Privacy-Preserving Machine Learning and Data Aggregation for Internet of Things

Lingjuan Lyu
ORCID: 0000-0003-3170-4994

Submitted in total fulfilment of the requirements of the degree of

Doctor of Philosophy

October 2018

Department of Electrical and Electronic Engineering
THE UNIVERSITY OF MELBOURNE

Abstract

The proliferation of *Internet of Things* (IoT) devices has contributed to the emergence of *participatory sensing* (PS) and *collaborative learning* (CL), where multiple participants collect and report their data to a cloud service to analyse the union of the collected data in the server-based framework. While in the decentralized framework, multiple participants collaboratively train a more accurate global model or multiple local models. However, the possibility of the cloud service or any participant being semi-honest or malicious pose a serious challenge of preserving the participants' privacy. Privacy-preserving machine learning and data aggregation aim to discover or derive useful statistics without compromising privacy.

This thesis systematically investigates state-of-the-art techniques for privacy-preserving machine learning and data aggregation in a range of IoT applications. Extensive theoretical and experimental results are provided to support the following primary contributions.

First, we explore three privacy-preserving machine learning applications. Examples include collaborative anomaly detection, human activity recognition and decentralized collaboration in a biomedical domain. We tackle security challenges in collaborative anomaly detection with a two-stage scheme called RG+RT: in the first stage, participants individually perturb their data by passing through a nonlinear function called *repeated Gompertz* (RG); in the second stage, the perturbed data are projected to a lower dimension using a participant-specific uniform random transformation (RT) matrix. The nonlinear RG function is designed to mitigate *maximum a posteriori* (MAP) estimation attacks, while random transformation resists *independent component analysis* (ICA) attacks. For human activity recognition, a similar two-stage scheme called RG+RP is proposed, the difference

lies in the second stage, where participants project their perturbed data to a lower dimension in an (almost) distance-preserving manner, using a *random projection* (RP) matrix. The random projection can both resist ICA attacks and maintain model accuracy. These proposed two-stage randomisation schemes are assessed in terms of their recovery resistance to MAP estimation attacks. Preliminary theoretical analysis as well as experimental results on synthetic and real-world datasets indicate that both RG+RT and RG+RP exhibit better recovery resistance to MAP estimation attacks than most state-of-the-art techniques, meanwhile high utility is guaranteed. To mitigate the inherent limitations in the centralized framework, and investigate the applicability of the decentralized framework, we study the decentralized collaboration in a biomedical domain. In particular, we develop an efficient *Decentralized Privacy-Preserving Centroid Classifier* (DPPCC) considering three practical scenarios, where *distributed differential privacy* (DDP) is combined with distributed exponential ElGamal cryptosystem to preserve privacy and maintain utility. We realize DDP using discrete Gaussian mechanism without any restriction on ϵ as in the traditional Gaussian mechanism, and only the encrypted noisy model parameters or test results are shared among all parties. It ensures each party learns nothing but the noisy sum of local statistics.

Second, we examine privacy-preserving data aggregation in smart grid application. To this end, we propose a multi-level aggregation framework based on fog architecture, which combines DDP with homomorphic encryption to perturb and encrypt smart meter data before forwarding to the nearby fog node. A two-layer encryption scheme is proposed to ensure aggregator obliviousness and system robustness.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Lingjuan Lyu, 15 October 2018

This page intentionally left blank.

Preface

This thesis reports on the body of work completed throughout the student's PhD research programme in The University of Melbourne. All the following work are supported by Melbourne International Research Scholarship (MIRS) and Melbourne International Fee Remission Scholarship (MIFRS) from The University of Melbourne. This thesis was mainly completed by the student. However, the student benefited from all collaborators through regular group meeting sessions or email discussions in which they provided valuable technical comments and guidance. The outcomes of this thesis are based on the following journals, conferences and drafts.

- Chapter 3

  -**Lingjuan Lyu**, Yee Wei Law, Sarah M. Erfani, Christopher Leckie, Marimuthu Palaniswami, "An improved scheme for privacy-preserving collaborative anomaly detection," *IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. IEEE, 2016, pp. 1-6.

  The contribution of each author is as follows: First author: proposing a novel non-linear function, performing all simulations, and writing the paper. Second author: providing technical support on the analysis of different attacks and revising the paper. Third, fourth, and fifth authors: proofreading, providing technical comments and supervision.

- Chapter 4

  -**Lingjuan Lyu**, Xuanli He, Yee Wei Law, and Marimuthu Palaniswami, "Privacy-preserving collaborative deep learning with application to human activity recognition," *Proceedings of the 2017 ACM Conference on Information and Knowledge Management*. ACM, 2017, pp. 1219-1228.

The contribution of each author is as follows: First author: proposing a two-stage scheme to revisit MAP and ICA attacks, and maintain recognition utility, performing all simulations, and writing the paper. Second author: proposing a LSTM-CNN model for human activity recognition, writing code for deep learning models. Third and fourth authors: paper revision, proofreading, providing technical comments, and supervision.

- Chapter 5

  -**Lingjuan Lyu**, Ben Rubinstein, Justin Bedo, Chris Culnane and Vanessa Teague, "Decentralized Privacy-Preserving Centroid Classifier," 2018. (draft under submission).

  The contributions of all authors (equal contributions) are as follows: Lingjuan Lyu: motivating the state-of-the-art research in decentralized collaboration by combining distributed differential privacy and encryption, performing all simulations, giving detailed analysis, and writing the paper. Ben Rubinstein: proving the stability of the discrete Gaussian distribution, removing the constraint of privacy budget $\epsilon < 1$ on the tradtional differential privacy. Justin Bedo: proposing SRU algorithm to feed floating-point values to encryption algorithm, inspiring the study of centroid classifier and different realistic scenarios. Chris Culnane: proposing distributed exponential ElGamal encryption algorithm to ensure party obliviousness and privacy, providing encryption implementation code. Vanessa Teague: pointing out weaknesses of current work, and proposing attacks against existing schemes.

- Chapter 6

  -**Lingjuan Lyu**, Karthik Nandakumar, Ben Rubinstein, Jiong Jin, Justin Bedo and Marimuthu Palaniswami, "PPFA: Privacy preserving fog-enabled aggregation in smart grid," *IEEE Transactions on Industrial Informatics*, 2018, pp. 3733-3744.

  The contribution of each author is as follows: First author: combining distributed differential privacy and encryption to derive multi-level aggregagtion of smart metering in a private manner, putting forward a two-layer encryption scheme to achieve aggregator obliviousness and authentication, performing all simulations, giving detailed analysis, and writing the paper. Second author: providing technical sup-

port on public-key cryptography for system robustness purpose. Third author: proposing to use the stability of Gaussian distribution to realize distributed differential privacy. Fourth author: recommending to use Fog computing for multi-level aggregagtion of smart metering. Fifth author: proposing SRU algorithm to feed floating-point values to encryption algorithm. Fifth author: supervision.

This page intentionally left blank.

Acknowledgements

This dissertation would not have been accomplished without the guidance and support of many nice people to whom I would like to express my sincerest gratitude in this acknowledgment.

First, I want to acknowledge my special thanks to my main supervisor Associate Professor Benjamin Rubinstein, for providing me constructive suggestions and help, he always taught me how to conduct more advanced research and encouraged me to be a better person.

I would like to thank my co-supervisor, Professor Marimuthu Palaniswami, for his insightful guidance and assistance through my PhD study.

I would also like to thank Professor Ampalavanapillai Nirmalathas for chairing my PhD committee. He gave me valuable comments, and advocated my research.

I am fortunate to be awarded with an IBM PhD fellowship, and have the chance to collaborate with many excellent scholars, including Prof.Jim Bezdek, Dr.Jiong Jin, Dr.Yee Wei Law, Dr.Justin Bedo, Dr.Vanessa Teague, Dr.Chris Culnane, Dr.Karthik Nandakumar, Dr.Sutharshan Rajasegarar, Yitong Li, Xuanli He, and many others. They are all quite helpful to my research career.

I cherish the opportunity of pursuing my PhD study in The University of Melbourne, and value all the people I encountered in Melbourne. Special thanks to my colleagues: Dr.Aravinda Sridhara Rao, Emerson Keenan, Fateme Fahiman, Shitanshu Kusmakar, Punit Rathore, Radhagayathri K. Udhayakumar, Mohammod Abdul Motin, Bigi Varghese Phillip, Nandakishor Desai, Shreyasi Datta, Wei Shen William Lim.

Finally, I want to express my deepest gratitude to my parents and family members for providing me with an exceptionally stimulating education environment. Their endless

love and never-failing spirit lead me to a bright life.

Contents

This page intentionally left blank.

List of Figures

List of Tables

This page intentionally left blank.

Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less.

Marie Curie

1 Introduction

1.1 Background

The proliferation of Internet of Things (IoT) technologies, mobile computing, and social network services provide the ability to capture data, either through compact, cheap, intelligent end nodes (sensors) or active contribution by people. An emerging data crowdsourcing paradigm is called *participatory sensing*(PS) [1] or *mobile crowdsensing* (MCS), where data are captured, either through sensors or active contribution by individuals with evermore capable mobile phones, and then sent to cloud services to analyze their collected data [2]. Meanwhile, in the real world, many applications would benefit from collaborative analysis across sensitive datasets maintained by different parties.

In these scenarios, participants want to extract information, improve local model, or learn knowledge from their joint records through machine learning or data aggregation techniques, without disclosing their privacy-sensitive information. Historically, Agrawal

and Srikant [3] were the first to study this problem, under the umbrella term "privacy-preserving data mining" (PPDM). PS/MCS promises novel applications in healthcare, fitness, citizen science, emergency preparedness, etc. For example,

In healthcare, thanks to the proliferation of wearable devices for measuring heart rate, blood pressure, movement, and many other physiological parameters, patients and elderlies can now be monitored through wearable devices, so that time-sensitive care can be given. Just as importantly, the wearable sensor data can be mined to reveal insights about the activities of the user community through *collaborative multi-user activity recognition* [4].

In fitness, integrated social networking drives data sharing, where there exists a large community whose members upload their wearable sensor data to online services that compile and publish community statistics. The ability to track and share fitness data online can enable positive reinforcement by peer groups, which encourages people to remain engaged in their physical activities and set new goals [5].

This trend is motivated by the need to collaborate with each other for more accurate analysis, and the fact that the data owned by a single party may be very homogeneous, resulting in an overfitted model that might deliver inaccurate results when applied to the unseen data, *i.e.*, poor generalizability. On the other hand, there is much demand to perform machine learning in a collaborative manner, since large amounts of data are often required to ensure sufficient computational power to test hypotheses as insufficient local training data may end up with worse models. In addition, decomposing and parallelizing computation among different parties could help reduce the demand for resources on any single party.

In terms of data aggregation, the energy, memory and computation constraints in distributed IoT end nodes pose various limitations that can make the network more vulnerable to malicious attacks and faults. The appearance of fog computing enables end nodes to send their data to the nearest fog node for secure multi-level aggregation, thus reducing time delay and resource bottlenecks.

1.2 Problem Statement

As participants' data commonly contain sensitive information, the increasing privacy and confidentiality concerns pose barriers to more widespread use of these private data in a broad of settings. The problem we are interested in can be stated as follows.

In centralized and distributed/federated machine learning frameworks, as illustrated in Figure 2.1 (a) and Figure 2.1 (b), the central server is expected to compute useful statistics or combine model information from multiple participants' data. While the central server is generally not malicious, it may be *semi-honest* or *honest-but-curious*, *i.e.*, it follows the protocol honestly, but may attempt to learn or infer sensitive information from users' data. In some domains, especially biomedical related applications, users' data is subject to great privacy risks. Moreover, users are also concerned that their data might be sold to third parties without their consent. For example, the fitness data collected by wearable devices, might include heart rate, location, calorie consumption, stress level, and other data that might reveal the user's identity, ethnicity, disease risks, and other sensitive information [6], hence users are reluctant to share their personal data, even learned model parameters or model updates.

Similarly, in decentralized machine learning, as depicted in Figure 2.1 (c), one of the participants might be malicious, who aims to sabotage the collaborative learning process or violate some of the privacy requirements by inferring information about the victim party's private data, which the attacker is not supposed to know. Consequently, data sharing and collaboration had been significantly hindered due to these privacy and confidentiality restrictions.

Another important privacy leakage examined by this thesis is in data aggregation. In particular, we study the privacy issues in smart grid, where the adversaries might eavesdrop on communications, compromise smart meters, or hack into the cloud or fog servers. The fine-grained energy consumption data can potentially be used to infer, profile and monitor household activity from the extracted information, for instance, the absence or presence of occupants, appliance usage, daily activities, even lifestyle of different households, and other such private information. In addition, the aggregator may have arbitrary auxiliary information a priori. These threats to security and privacy have been broadly studied [6–10]. Meanwhile, these privacy concerns could be largely reduced if

only their aggregation is released, while still providing the benefits of data cooperation for operational purposes. For example, individual privacy leakage can be reduced by releasing the aggregate of household usage information. However, there is no guarantee that the resulting aggregate ensures privacy as it still contains private information. This holds even more for a daily profile of the aggregate values [11].

This thesis aims to address the above privacy issues in machine learning and data aggregation, by investigating a range of relevant applications in Section 1.3. In particular, this thesis assumes honest-but-curious threat models for privacy-preserving machine learning applications in Chapter 3, Chapter 4, and Chapter 6, while privacy-preserving data aggregation in Chapter 5 considers 2/3 parties to be honest-but-curious.

1.3 Research Applications

For privacy-preserving machine learning, we explore three practical applications. In particular, for centralized privacy-preserving machine learning, we study *Privacy-Preserving Collaborative Anomaly Detection*, and *Privacy-Preserving Collaborative Human Activity Recognition*. For decentralized privacy-preserving machine learning, we investigate a practical application in the biomedical domain, *i.e.*, *Decentralized Privacy-Preserving Machine Learning in Biomedical Domain*. For privacy-preserving data aggregation, we propose *Privacy-Preserving Fog-enabled Aggregation in Smart Grid*.

- **Chapter 3: Privacy-Preserving Collaborative Anomaly Detection:** The process of detecting interesting or unusual events in the network is known as anomaly detection [12]. Anomaly detection is important for tasks such as security, fault diagnosis, intrusion detection, and monitoring applications. Given the possibility that the cloud service is honest but curious, a major challenge is *how to protect participants' privacy*.

- **Chapter 4: Privacy-Preserving Collaborative Human Activity Recognition:** We consider the scenario where a cloud service provider performs deep learning on the collected data contributed by a large number of participants. To remove the deterrents for users to share their data and benefit from the community knowledge discovery afforded by *participatory sensing* (PS) or *mobile crowdsensing* (MCS), service

providers need privacy-preserving algorithms that deliver a reasonable trade-off between data privacy and utility. Hence the problem we are addressing is *how a data owner can release its data with the guarantees that the original sensitive information cannot be recovered while the analytic utility of the data are preserved*.

- **Chapter 5: Decentralized Privacy-Preserving Machine Learning in Biomedical Domain:** A wealth of practical applications fall under horizontally partitioned database category, where multiple parties each manage different groups of individuals with similar features. For example, in biomedical domains, various types of data (*i.e.*, demographic, clinical, and genomic) are increasingly being digitized, collected and stored in independent hospitals or biomedical research institutions [13]. However, local repositories or institutions are reluctant to share their data with the susceptible centralized third party due to privacy concerns and in some jurisdictions, legislation. This prevents researchers and consequently patients from benefiting from distributed data using machine learning techniques. Therefore, we are motivated to develop a decentralized interactive solution by removing the trusted server under the assumption that strictly less than one third of the parties are malicious. On the other hand, because a learning model trained on some dataset can be considered as a high level statistic for that dataset, and even the predication of a trained model can reveal privacy about training data through black-box attack [14–16]. Therefore, we consider how to facilitate decentralized collaboration by sharing locally learned model parameters or model outputs in a privacy-preserving manner. In particular, we apply the efficient *Decentralized Privacy-Preserving Centroid Classifier* (DPPCC) to three practical cases, where only the encrypted noisy model parameters or model predictions are shared among all parties. DPPCC provides an efficient, and privacy-preserving solution for collaboration among multiple hospitals or biomedical research institutions.

- **Chapter 6: Privacy-Preserving Fog-enabled Aggregation in Smart Grid:** Smart grids have emerged and been recognised as the next generation of power grid [17], and is expected to provide improved reliability, flexibility, sustainability, consumer involvement, and security. As an essential feature of smart grids, smart metering allows residential customers to monitor, track and reduce electricity costs through

smart meters (SMs). State-of-the-art information and communication technologies facilitate communication between grid participants and the service provider (aka smart grid operation center in this context), which aims to compute useful statistics to serve for various social and economic purposes, as well as deliver better services, *e.g.*, load modeling/forecasting/management, monitoring real-time consumption, network planning and settlement, energy generation and price optimization as per demand. Furthermore, users will be rewarded by changing their consumption habits to better match generation [7]. With the prevalence of smart appliances, it is expected that the smart grid can even remotely schedule the selected appliances to flatten demand [8]. As an important aspect in smart grids, the sum/aggregation of energy consumption data at multiple levels (neighborhood, subdivision, district, city, *etc*.) is essential for load modeling/forecasting/management, monitoring real-time consumption, predicting power consumption, network planning and settlement, energy generation and price optimization as per demand, *etc*. [9, 10, 18–21]. Hence, the key problem in the multi-level aggregation of smart metering is *how to minimize the privacy leakage of the locally released statistics while ensuring high utility*.

The chapters of this thesis explore different settings of centralization and different applications. In many of these cases different privacy/utility trade-offs can be struck, and so we have applied a range of privacy frameworks and techniques as case studies. For all the four applications in this thesis, Table 1.1 summarizes the key differences from the perspective of frameworks, threat models, privacy mechanisms and privacy metrics.

We want to remark that although the first stage in the two-stage schemes of Chapter 3 and Chapter 3 both share the same nonlinear RG function, the second stage is different to tailor to different applications. Similarly, it should be noted that although Chapter 5 and Chapter 6 both adopt DDP and homomorphic encryption, there are at least two obvious differences: (1) Chapter 5 puts forward a discrete Gaussian mechanism to achieve (ϵ, δ)-differential privacy without restriction on ϵ, via stability of the discrete Gaussian distribution implemented through rejection sampling. In contrast, Chapter 6 directly follows the traditional (ϵ, δ)-differential privacy [22], where $\epsilon < 1$; (2) The encryption scheme and realization are totally different. Chapter 5 develops a distributed exponential ElGamal cryptosystem to support fault tolerance, availability of our protocol, etc. In

Table 1.1: Differences between the four applications in this thesis.

Chapter	Applications	Frameworks	Threat models	Privacy mechanisms	Privacy metrics
3	Anomaly detection	Centralized	Honest-but-curious	Data randomization (RG+RT)	Recovery rate
4	Human activity recognition	Centralized	Honest-but-curious	Data randomization (RG+RP)	Recovery rate
5	Biomedical domain	Decentralized	2/3 parties to be honest-but-curious	DDP + OTP + Public key cryptosystem	(ϵ, δ)-DP ($\epsilon < 1$)
6	Smart grid	Fog-empowered multi-level	Honest-but-curious	DDP + Distributed exponential ElGamal cryptosystem	(ϵ, δ)-DP (remove constraint on $\epsilon < 1$)

contrast, Chapter 6 proposes a two-layer encryption scheme: the first layer applies OTP to ensure aggregator obliviousness, while the second layer uses public-key cryptography for authenticity. These differences are highlighted in Table 1.1.

1.4 Contributions of this Thesis

This thesis systematically studies privacy-preserving machine learning and privacy-preserving data aggregation, and explores four practical applications in IoT, including collaborative anomaly detection, human activity recognition, decentralized collaboration in biomedical domain and smart metering. It delivers the following contributions in four main chapters.

- **Chapter 3** tackles the security challenges in collaborative anomaly detection via a cloud server. Our contributions here are two-fold: (i) the proposal of a two-stage scheme called RG+RT: in the first stage, participants perturb their data by passing through a nonlinear function called *repeated Gompertz* (RG); in the second stage, the perturbed data are projected to a lower dimension using a participant-specific uniform *random transformation* (RT) matrix. The nonlinear RG function is designed to mitigate *maximum a posteriori* (MAP) estimation attacks, while random transformation aims to resist *independent component analysis* (ICA) attacks, and (ii) the evalua-

tion of the privacy-preserving properties of the existing schemes and the RG+RT. In terms of the recovery rate, which measures how well an attacker can recover the original data from the perturbed data, RG+RT proposed here is found to be more resistant to MAP estimation attacks than the state-of-the-art schemes.

- **Chapter 4** addresses privacy issues in human activity recognition. Our contributions here are as follows: (i) the proposal of a two-stage perturbation mechanism called RG+RP for privacy-preserving collaborative deep learning, which maintains the privacy of both normal and anomalous records in terms of resistance to both MAP estimation attacks and ICA attacks. The random projection stage ensures the accuracy of time-series classification, especially for large data size. In addition, RP also contributes to energy saving transmission by compressing data; (ii) a novel LSTM-CNN model is applied to human activity recognition from wearable sensor data. LSTM-CNN model demonstrates better performance than the traditional DBN model and standalone LSTM, CNN models, and preserves learning accuracy even after the two-stage perturbation. Preliminary theoretical analysis as well as experimental results on synthetic and real-world datasets indicate that RG+RP exhibit better recovery resistance to MAP estimation attacks than most state-of-the-art techniques. Our solution shows great promise for building MCS applications that respect the participants' privacy.

- **Chapter 5** examines the limitations in the server-based framework, and investigates the possibility of a decentralized framework to facilitate decentralized collaboration among different institutions in the biomedical domain. For this chapter, we make the following main contributions:

  (i) We address centroid classifier learning without central aggregation, under a strong threat model of Byzantine fault tolerance;

  (ii) We provide theoretical analysis and investigate three practical sharing cases where all parties collaborate, without disclosure of their data, to:

  1. learn a shared model;
  2. produce a prediction for a given sample and;
  3. produce a prediction without revealing the sample or prediction.

(iii) We put forward a discrete Gaussian mechanism to achieve (ϵ, δ)-differential privacy without restriction on ϵ, via stability of the discrete Gaussian distribution implemented through rejection sampling; and

(iv) We conduct a comprehensive experimental study on biomedical datasets, confirming the practicality of our framework.

- **Chapter 6** deals with privacy concerns in smart metering. Our contributions here include: (i) we propose an efficient and privacy-preserving aggregation model with fog/cloud hybrid architecture. It povides multi-level aggregation, and achieves aggregator obliviousness. In particular, the intermediate fog nodes can periodically collect data from the connected smart meters and derive fine-grained fog level aggregation for further cloud level aggregation, thus conserving communication energy. (ii) we combine DDP with additive homomorphic encryption to derive differentially-private aggregation; (iii) to mitigate privacy loss and maintain utility, a two-layer encryption scheme is proposed: the first layer applies one-time pad (OTP) to encrypt individual noisy measurement in an additive homomorphic manner to achieve aggregator obliviousness, while the second layer provides authentication by using public-key cryptography; (iv) our privacy model is stronger in the sense that we do not trust the aggregator. We ensure that the aggregator is only able to learn the desired statistics without knowing additional information. Furthermore, our model supports authenticity, malfunction/non-responding nodes, and node joins and departures; (v) we give theoretical analysis, and conduct a comprehensive study on real-world smart metering data, confirming the superiority of our proposed schemes.

1.5 Thesis Organization

This thesis is organized as follows. Figure 1.1 provides an overview of the organization of this thesis.

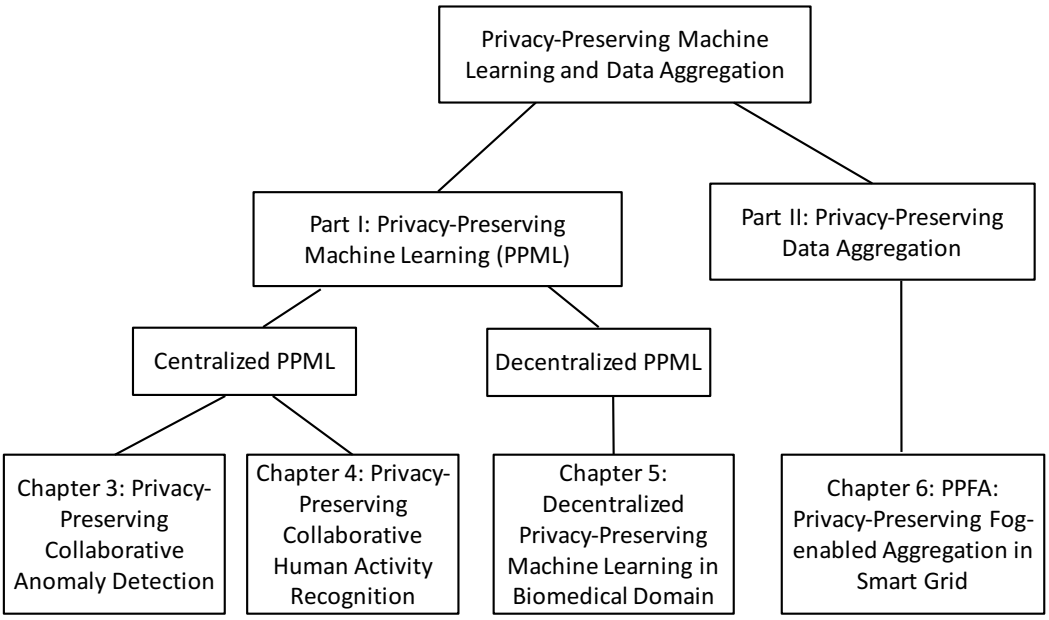


Figure 1.1: Thesis organization.

- **Chapter 1**: This chapter presents the background of this research, the problem statement, research applications, thesis contributions, and the organization of this thesis.

- **Chapter 2**: This chapter offers a complete overview of various frameworks, adversary models, privacy attacks and techniques that have been developed for privacy-preserving machine learning. This chapter first presents different machine learning frameworks, followed by the review of the relevant privacy issues and existing solutions. Fog-empowered data aggregation is introduced to address the security issues in smart grid. Next, it reviews popular adversary models and privacy attacks, followed by the privacy-preserving techniques that consist of encryption, data randomisation, and differential privacy.

  Part I: This part explores centralized and decentralized privacy-preserving machine learning applications in seperate chapters, including *Privacy-Preserving Collaborative Anomaly Detection*, *Privacy-Preserving Collaborative Human Activity Recognition*, and *Decentralized Privacy-Preserving Machine Learning in Biomedical Domain*.

- **Chapter 3**: This chapter focuses on privacy-preserving collaborative anomaly detection, a two-stage scheme called RG+RT is proposed: in the first stage, a nonlinear function called *repeated Gompertz* (RG) is used to perturb each participant's

data, and mitigate *maximum a posteriori* (MAP) estimation attacks; in the second stage, each participant uses a specific uniform *random transformation* (RT) matrix to resist *independent component analysis* (ICA) attacks. The proposed two-stage randomisation scheme is assessed in terms of its recovery resistance to MAP estimation attacks.

- **Chapter 4**: This chapter discusses how to perform accurate collaborative human activity recognition without compromising privacy, and studies the basic distance-preserving properties of multiplicative perturbation techniques. RG+RP is applied to mitigate *maximum a posteriori* (MAP) estimation attacks, resist *independent component analysis* (ICA) attacks and ensure model accuracy. A novel LSTM-CNN model is developed for human activity recognition.

- **Chapter 5**: This chapter progresses the decentralized collaboration in biomedical domain. An efficient *Decentralized Privacy-Preserving Centroid Classifier* (DPPCC) is developed and three practical scenarios are considered, where distributed differential privacy is combined with distributed exponential ElGamal cryptosystem to preserve privacy and maintain utility. A discrete Gaussian mechanism is put forward to achieve (ϵ, δ)-differential privacy without restriction on ϵ, via stability of the discrete Gaussian distribution implemented through rejection sampling

  Part II: This part studies the problem of privacy-preserving data aggregation in smart grid application, *i.e.*, how to derive accurate multi-level data aggregation, reduce transmission energy, and preserve user privacy.

- **Chapter 6**: This chapter addresses the security issues in smart grid, and focuses on how to aggregate smart meter data in a privacy-preserving manner. We combine DDP with efficient homomorphic encryption to perturb and encrypt smart meter data, and provide multi-level aggregation using fog/cloud hybrid framework. A two-layer encryption scheme is presented to ensure aggregator obliviousness and system robustness, including authenticity, node failure, node joins and departures.

- **Chapter 7**: This chapter concludes this thesis and outlines the directions for future research.

This page intentionally left blank.

Those who tell the stories rule the world.

Plato

2

Literature Review and Related Work

2.1 Database Partitioning

The most common type of data storage is relational databases [23], which consists of multiple records/tuples, and each record/tuple contains multiple features/attributes. In distributed collaborative learning, relational datasets belonging to different data owners can be categorised into the following three categories:

- **Horizontally partitioned database:** different parties hold different records containing the same attributes [24, 25];
- **Vertically partitioned database:** different parties hold different attributes of the same records [26–28];
- **Arbitrarily partitioned database:** for each record, attributes are arbitrarily distributed across multiple parties [29, 30].

In the following chapters, we mainly focus on horizontally partitioned database, where training data comprising m training examples are split over a matrix $\mathbf{X} \in \mathbb{R}_{m \times d}$ on d features and associated labels $\mathbf{Y} \in \mathbb{R}_m$. The whole database is partitioned into n disjoint subsets $(\mathbf{X_i}, \mathbf{Y}_i)$ for $i \in \{1, n\}$ representing n parties each curating their own part of the data over a common d-dimensional feature space.

2.2 Machine Learning Frameworks

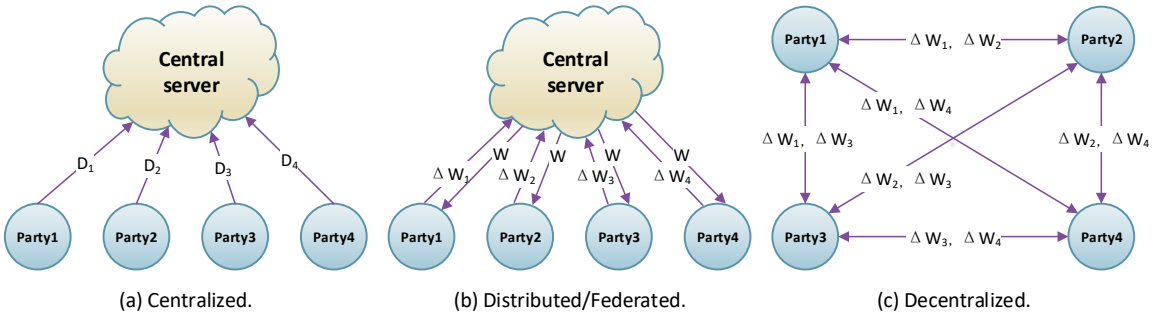
In general, the frameworks for machine learning mainly fall into the following three categories: centralized framework, distributed/federated framework and decentralized framework.

- **Centralized framework**. As illustrated in Figure 2.1 (a), parties do not need to learn any models in the centralized framework. Instead, parties send their data to a centralized server, who is entitled to see all parties' data in the clear, and learn a global model to provide beneficial services (*e.g.*, prediction API, video editing tools or health apps, etc).

- **Distributed/Federated framework**. As illustrated in Figure 2.1 (b), parties participate in the learning process, while the server is entitled to mediate the modeling process by aggregating local models in the distributed/federated framework.

- **Decentralized framework**. However, due to privacy sensitivity, a party may not trust a single central server [31], and so may not be willing to transfer its data or model parameters to the server. Moreover, the central servers in both centralized framework and distributed/federated framework are susceptible to single-point-of-failure, single-point-of-breach, etc, hence a decentralized implementation is highly desired, as illustrated in Figure 2.1 (c).

Below, we review the relevant privacy issues and existing solutions for different machine learning frameworks.



(a) Centralized. (b) Distributed/Federated. (c) Decentralized.

Figure 2.1: Different machine learning frameworks.

2.2.1 Centralized Machine Learning

Most of the centralized machine learning frameworks force multiple participants to pool their data into a trusted centralized server to train a global model on the combined data, as depicted in Figure 2.1 (a). This centralized framework is very effective, but it's not privacy-preserving since the server has direct access to all sensitive data. Moreover, a trusted server rarely exists in real world, as the excessive data collected from users could be used for unintended purposes (*e.g.*, face recognition for targeted social advertising), posing immediate or potential privacy risks. As pointed out by Shokri *et al.* [32], centralized learning poses serious privacy threats: (i) all the sensitive training data are revealed to a susceptible third party who can keep the collected data forever; (ii) data owners have no control over the learning objective or knowledge of what can be inferred from their data; (iii) the learned model is not directly available to data owners. To mitigate these privacy risks, Microsoft developed CryptoNets [33,34] to perform learning on encrypted data and provide encrypted outputs to users [33]. However, CryptoNets assumes the weights of the neural network have been trained beforehand, and aims at making prediction for test records. Another alternative is to perturb data before forwarding to the server, while maintaining the analytic properties [35–38]. Chapter 3 and Chapter 4 study two practical centralized machine learning applications, and address two representative privacy attacks via data randomization.

2.2.2 Distributed/Federated Machine Learning

In contrast, in distributed/federated learning, participants are involved in the training process, and the server is entitled to distribute key shares, mediate the modeling process or aggregate local models, as illustrated in [8–10, 32]. Wainwright *et al.* [39] introduce the concept of distributed deep learning to protect privacy of local training data, as illustrated in Figure 2.1 (b). Instead of explicitly sharing training data, parties collaboratively train a global model by sharing only a fraction of the gradients of their individually trained local models with each other through the mediate of parameter server. Distributed learning is further explored in [32, 40–42]. Mohassel *et al.* [43] provide a solution for training neural networks while preserving privacy of participants. However, they deploy *secure multiparty computation* (SMC) in the two-server model where clients outsource their computation to these two servers who are assumed not colluding with each other. In general, SMC techniques achieve a high level of privacy and accuracy, at the expense of high computational and communication overhead for the participants, thereby doing a disservice to attracting participation.

Alternatively, Shokri *et al.* [32] are the first to introduce the concept of *Distributed Selective Stochastic Gradient Descent* (DSSGD) as an alternative to the costly SMC techniques. It allows each party to keep local model private while iteratively update its model by integrating differentially-private gradients of other parties through a parameter server. Communication cost is addressed by only sharing a fraction (*e.g.*, 1%-10%) of local model gradients that are above a threshold or the gradients with the largest absolute values through a parameter server during each round of communication. To achieve this goal, they exploit the fact that optimization algorithms, such as SGD, can be parallelized and executed asynchronously. Each party computes local model gradients based on local training data, then a fraction of gradients are forwarded to parameter server, who is honest-but-curious: it is assumed to be curious in extracting the data of individuals; and yet, it is assumed to be honest in operations. Their approach includes a selective parameter sharing process combined with updating local model parameters during SGD. Each participant uploads and downloads a percentage of the most recent updates to avoid getting stuck into local minima. In addition, the shared local model gradients are blurred by adding noise using differential privacy. However, besides all the weaknesses in the

server-based model as mentioned in Section 1.2, their model poses the following risks:

- **Meaningless privacy**: Their privacy bounds are given per-parameter, the large number of parameters prevents the technique from providing a meaningful privacy guarantee. For example, Shokri reports $\sim$ 92% accuracy on SVHN with privacy budget $\epsilon > 2$ per epoch of 300,000 model parameters, naively, this corresponds to a total $\epsilon > 600,000$ for $\theta_u = 1$, and $\epsilon > 60,000$ for $\theta_u = 0.1$, which is actually meaningless for differential privacy as the total privacy loss per participant exceeds several thousand [44].

- **Privacy leakage**: Because DSSGD offers meaningless privacy, as demonstrated by [45], local data information may be leaked to an honest-but-curious parameter server, even a small portion of original gradients may reveal information of local data, and thus the server can extract individual data with non-negligible probability in the case of only one neuron, and even for general neural networks with regularization, the gradients can still reveal the label of the original data.

- **Vulnerability to GAN attack**: Another assumption in the server-based framework is that all the parties are honest, neglecting the possibility of malicious parties. In reality, if a party turns out to be malicious, it can easily sabotage the learning process (*e.g.*, by spoofing random data samples) or violate some of the privacy requirements by inferring information about the victim party's private data, which the attacker is not supposed to know. Hitaj *et al.* [46] devise an active inference attack on deep neural networks in a collaborative setting, which is referred to as *Generative Adversarial Networks* (GAN) attack. It exploits the real-time nature of the learning process that allows the adversarial party to train a GAN that generates prototypical samples of the targeted training data that was meant to be private and the generated samples are intended to come from the same distribution as the training data. Figure 4 in [46] illustrates how a malicious party can intentionally compromise any other party via server, the malicious party is able to operate successfully as long as the global model is under the process of learning. GAN attack makes the distributed setting even more undesirable, as in centralized learning only the server can violate users' privacy, but in distributed learning, any user may violate the pri-

vacy of other users in the system, even without involving the server [46]. It is worth noting that GAN attack succeeds only if the following three conditions are held: (i) adversary has knowledge of labels of other participants; (ii) class distribution of the adversary and honest parties are non-IID; (iii) each party does not adopt any privacy protection mechanism or offers meaningless privacy.

DSSGD was later extended to federated learning to deal with non-IID and unbalanced data, including FedAvg and FedSGD developed by researchers from Google [41,47]. The goal is to train a shared model while leaving training data on users' mobile phones. Instead, mobile phones with relatively powerful and fast processors (including GPUs) are required to download the current model, compute an update by performing local computation, then send local model update to the trusted Google Cloud server in each epoch of training process. To protect individual model updates from the adversarial server who might scrutinize individual updates in federated learning, instead of using differential privacy as in [32], Bonawitz *et al.* [47] propose a secure aggregation protocol to protect privacy of model updates of each user. The updates from individual users are securely aggregated by SMC as the weighted average of model gradients. Another more efficient method is to borrow differential privacy to guarantee user-level privacy, as demonstrated by McMahan *et al.* [42]. However, the default trusted Google server is entitled to see all users' update in the clear, aggregate these updates and add noise to the aggregation, resulting in privacy leakage, hence their scheme is even weaker than DSSGD when the server is untrusted.

2.2.3 Decentralized Machine Learning

It has been pointed out that the central server-based learning frameworks, including centralized and distributed/federated machine learning frameworks, suffer from the following weaknesses:

- **Untrusted server**. Due to privacy sensitivity, a party may not trust a single central server [31], and so may not be willing to transfer its data or model parameters to the server.

- **Single-point-of-failure**. If the central server is shut down for maintenance, the

whole network stops working [48]. Moreover, if the central server is attacked, the entire network is under the risk of being compromised [31].

To address and overcome these weaknesses, a practical solution is to remove the central server and parallelize computation among multiple parties to help reduce the demand for resources on any single party, as shown in Figure 2.1 (c). The decentralized framework is much stronger than server-based frameworks, in the sense that it is a purely decentralized framework without relying on any central server. The first decentralized machine learning model is ModelChain [49], which applies Blockchain technology to machine learning by incorporating the idea of boosting, *i.e.*, samples that are more difficult to classify are more likely to improve the model significantly. More specifically, the global model is initialized with the local model with the lowest error to prevent error propagation, and in the follow-up epochs, the party with the highest error is chosen to be the winner party to update the model as it contains the most information to further improve the model, and thus should be assigned a higher priority to be chosen as the party to update the model. The updating process is repeated until the consensus global model is derived, *i.e.*, when a party wins the update bid in two consecutive epochs. However, their proposed logic for ModelChain is reasonable only if all the participants are completely honest. Furthermore, privacy issues are not considered. In their ModelChain, each party publicly reveals its plain model rather than publishing less sensitive model parameters, thus each party can get access to the intermediate and consensus global model. Another more reasonable choice is to apply differential privacy in the decentralized framework, which gives rise to the concept of *distributed differential privacy* (DDP) to reflect the fact that the noise in the target statistic is sourced from all participants. Dwork *et al.* [50] first described a distributed protocol to achieve differential privacy in the decentralized framework: the shares of random binomial noise are generated by performing coin flipping, which is secure against malicious participants, however, it requires communication among users and the expensive secret sharing technique results in $O(n)$ multiplications and additions in shares, where n denotes the number of participants, thus not preferable for large number of participants. In Chapter 5, we address the privacy issues in the decentralized machine learning for three practical scenarios in biomedical domain.

2.3 Fog-Empowered Data Aggregation

With the growing quantity of data generated at the edge of the IoT network, the speed of data transmission is becoming a bottleneck in the cloud computing paradigm. For example, some IoT applications require fast timely responses, some involve private data, and some produce large amounts of data, which might cause serious communication burden on the network. The difficulties in scaling to ever increasing numbers of sensors and actuators, unpredictable response time from cloud server to end nodes and unreliable cloud connections can bring down the services offered by cloud, especially for large-scale applications, such as smart grids. Additionally, the excessive data collected from end nodes could be used for unintended purposes, posing immediate or potential privacy risks. Therefore, it can be beneficial to forward data to the nearby one-hop away fog nodes. In this regard, fog architecture serves as a potential technique to provide off-loading for the cloud, and reduce the time delay and energy cost within an IoT network [51].

The fog paradigm is well positioned for real-time large-scale data analysis by supporting densely distributed data sources, and providing superior user experience [52, 53]. It extends cloud computing services to the edge of the network by providing users with computation, storage, and application services. Fog nodes can be distinguished from cloud servers by their proximity to end-users, dense geographical distribution and support for mobility [54]. In addition, fog architectures are potentially more privacy friendly than cloud architectures. Henceforth, fog computing becomes an efficient alternative to cloud computing in many IoT applications. In general, while the cloud has the largest storage capacity and the highest processing capability, fog nodes have intermediate storage and processing capabilities, and end nodes, which are often battery powered, have limited memory, processing and energy resources. Consider a Libelium Waspmote (sensor node) with a 14.7456MHz ATmega1281 processor, 8KB SRAM, 4KB EEPROM, and 128KB flash memory. It can communicate at 2.4GHz frequency (250Kb/s) using an XBee module [55], and store at most 100 two-dimensional floating-point number.

As estimated by Cisco Global Cloud Index, by 2019, data produced by people, machines, and the IoT will reach 500 zettabytes, however, the global data center IP traffic will only reach 10.4 zettabytes by that time [56]. Further, 45% of IoT-created data will be stored, processed, analyzed, and acted upon close to, or at the edge of the network

[57]. Fog nodes can be effectively utilized to perform data processing, fine-grained aggregation or timely anomaly detection. Furthermore, processing on the nearby fog nodes facilitates early detection at the source, and prolongs the lifetime of the energy-limited network. Consequently, we provide a promising solution based on fog computing architecture, which could be used for large-scale IoT applications.

We envision a Fog-empowered multi-level framework (three-level for illustration purpose) in Figure 2.2, there exist numerous end nodes at the bottom layer, fog nodes at the middle layer and cloud at the top layer. The fog nodes at the middle layer can act as both fog server with more powerful storage and processing capability, and fog edge nodes, who can interact with heterogenous devices, such as different types of end devices with various protocols. Therefore, a fog node can process/aggregate measurements from nearby end devices through various communication protocols, e.g. Wi-Fi, BlueTooth, ZigBee, and cellular network, and forward the results to the remote cloud [52]. The commercially available products for a fog node candidate can be Intel Edison, Raspberry Pi, Arduino Uno, and other microcontroller boards [52].

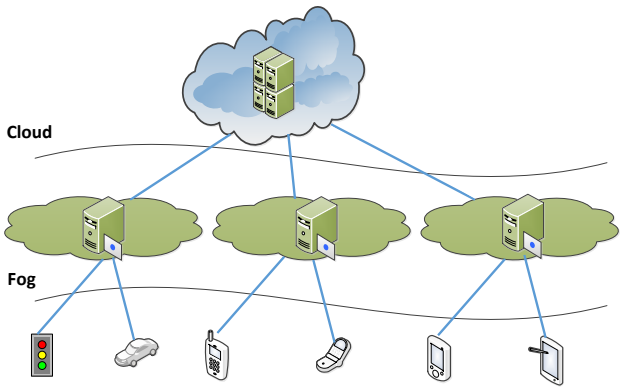


Figure 2.2: Fog-empowered multi-level framework.

Considering the above benefits offered by fog architecture, we investigate how to use fog/cloud hybrid architecture for privacy-preserving aggregation of smart metering. Given a daily profile, spatially aggregated over a total of n smart meters, an important challenge is how to perform multi-level aggregation in a privacy-preserving manner that respects data confidentiality without sacrificing utility, especially when the data aggregator is untrusted. In order to support this application, cloud computing alone is not efficient enough as the requirements of privacy protection and the huge communication

cost pose obstacles in cloud. In the scenario where the remote cloud is only concerned with the overall aggregation of all regions over time, the individual measurements in each region can be aggregated at the nearby fog node and then privately forwarded in a compact form to the cloud. Hence, fog-empowered data aggregation in Chapter 6 aims to explore how to derive multi-level aggregation, reduce transmission energy, and preserve user privacy.

2.4 Adversary Models

- Semi-honest: In this case, the adversary is considered to be passive or honest-but-curious, who tries to learn the private states of other parties, without deviating from the protocol.
- Malicious: In this case, an active attacker tries to learn the private states of other parties, and deviates arbitrarily from protocol by modifying, re-playing, or removing messages. This strong adversary model allows the adversary to conduct particularly devastating attacks.

2.5 Privacy Attacks

Privacy and confidentiality concerns pose barriers to more widespread use of private data. For example, the private data collected from users could be used for unintended purposes (*e.g.*, face recognition, location-based services, or mobile health applications for targeted social advertising and recommendation), posing the immediate or potential privacy risks, hence private data should not be directly shared without any privacy consideration. Even the trained models or model predictions can reveal data privacy [14, 58] as models with high capacity can simply memorize the training data, especially the modern deep learning models have vastly more capacity than they need to perform well on their tasks, as evidenced by the following privacy attacks.

1. ***Model inversion attack***: Given a model and a known output, the model inversion attack tries to infer the values of sensitive attributes. To this end, the attacker trains a machine learning model that learns the hidden features by following the

gradients of the error with respect to the hidden input features so as to minimize the error between the predicted output and the known output. As a result of this reverse-engineering, the attacker can recover prototypical examples from the training set. Fredrikson et al. [59] demonstrated two cases: one case study of model inversion attack was on pharmacogenetics analysis, where the correlation between a patient's genotype and the dosage of a certain medicine were captured by training an adversarial model to infer disease information about the patient; another case study was in reconstructing the face of an individual in a face recognition system, provided that the particular face label is given. In fact, it has been shown that these prototypical examples simply correspond to the average of the input features that characterize the corresponding output, *i.e.*, class representatives, but do not represent any actual member of the training set [14].

2. ***Model extraction attack***: A privacy breach occurs if an adversary can reconstruct/steal model in practice through prediction APIs, which is known as model extraction attack [58]. In specific, an adversary extracts important parameters of a model that is trained on private data, and uses these parameters to train an adversarial model to mimic the performance of the original model.

3. ***Membership inference attack***: Another recent attack called membership inference attack is proposed by Shokri et al. [14], which ideally applies to any type of machine learning models. Given an input record and black-box access to the target model, the aim of an adversary is to determine if the input record comes from the training data of the target model or not, *i.e.*, its membership. In particular, an attack model is trained to recognize differences in the target model's predictions on the inputs that it trained on versus the inputs that it did not encounter during training. The attack model performs binary classification on a data record, which indicates whether this data record belongs to the training set of the target model or not, depending on the class of the data record and the output of the target model on it. Also the attack model is itself composed of a set of models, one for each of the output classes of the target model. To train the attack model, shadow models are built which are of the same architecture and type as of the target model, hence are close in behaviour to the target model.

4. ***Secrete extraction attacks***: Orthogonal to membership inference, Carlini *et al.* [15] showed that recurrent language models trained on text data are capable of memorizing unique patterns in the training data, and an adversary can reconstruct specific training points (a random secret number) with black-box access to the model given some prior knowledge about the format of the input (*e.g.*, credit card number). For a text generation model, numbers are essentially random data, thus this illustrates that models can memorize random data. Moreover, Song *et al.* [60] showed that text-generation models can memorize even words and sentences that are directly related to their primary task without a negative impact on the test accuracy.

Therefore, neither the private training data nor the derived model or model prediction shall be directly shared with any untrusted parties, motivating the relevant works in this thesis.

2.6 Privacy-Preserving Techniques

There exists a growing body of literatures on *semantic* privacy-preserving techniques, which are concerned with minimising the difference between adversarial prior and posterior knowledge about the individuals represented in a database. This section reviews the current privacy-preserving methodologies that have been developed in this area, including encryption, randomization, and differential privacy.

2.6.1 Privacy-Preservation through Encryption

There exist cryptographic techniques that allow algorithms to be modified to process encrypted data, these techniques fall under the heading of *secure multiparty computation* (SMC) [61]. SMC ensures a high level of privacy and accuracy, but the main challenge facing SMC-based schemes is the requirement for simultaneous coordination of all participants during the entire training process, which limits the number of participants.

On the other hand, using public key cryptography in all communications is costly and time consuming, which is unacceptable for resource limited end nodes. Considering computational complexity of asymmetric encryption and efficiency of symmetric encryption, one alternative solution is to use public key cryptography to transfer the symmetric

encryption key, which is actually used to encrypt the released statistics. In such a cryptosystem, a shared secret key (pseudo-random session key) is generated in advance, and this much briefer session key is then encrypted by the recipient's public key. Only the recipient can use the corresponding private key to obtain the session key, then use the session key to decrypt the received ciphertext.

For aggregation tasks, the additive homomorphic property is desired. A homomorphic encryption scheme allows arithmetic operations to be directly performed on ciphertexts, which is equivalent to a specific linear algebraic manipulation with the plaintext. It is especially useful when someone who does not have the corresponding decryption key needs to perform arithmetic operations on a set of ciphertexts. For example, for fog-empowered aggregation, fog level and cloud level aggregation can be obtained by decrypting the sum of ciphertexts using additively homomorphic encryption. The fully homomorphic encryption focuses on homomorphic operations on ciphertexts encrypted under the same key [62], which is inapplicable for aggregation purpose. Because if users encrypt their data with the aggregator's public key, the aggregator could gain both the aggregate statistics and each user's plaintext. Well-known partially homomorphic encryption schemes include: RSA [63], El Gamal [64], Paillier [65], etc. Specifically, we implement additively homomorphic encryption using exponential ElGamal [66] in Chapter 5 and efficient stream cipher [19] in Chapter 6.

2.6.2 Privacy-Preservation through Randomization

To dispense with the additional, high-overhead cryptographic mechanisms, most randomization-based schemes use alternative semantic privacy criteria/definitions. For *Random Multiparty Perturbation* (RMP), we use the criterion called *recovery resistance*. This criterion is based on the *recovery rate* metric used in Sang *et al.*'s innovative study of attacks on randomization-based schemes [67].

In general, randomization techniques include (i) additive perturbation, (ii) multiplicative perturbation, (iii) geometric perturbation, and (iv) nonlinear transformation.

Additive perturbation adds independent and identically distributed (i.i.d.) noise to the original data [3], but Kargupta *et al.* [68] and Huang *et al.* [69] questioned the use of random additive noise and pointed out that additive noise can be easily filtered out in

many cases. The general principle is that the additive noise should be correlated with the data [70], but applying this principle to data that is not time-series is problematic, because the additive noise can be filtered to a good approximation of the original data [69]. Moreover, a participant can infer the data of another participant if their data happen to be correlated [71].

Multiplicative perturbation premultiplies the original data with a random noise matrix. Well-known multiplicative schemes include:

- *Rotation perturbation* defines the noise matrix as a matrix with orthonormal rows and columns [72]. This scheme is vulnerable to "known-input attacks" [73], where an attacker can recover the original data from its perturbed version with just a few leaked inputs.

- *Random projection* (abbreviated as RP) leverages the *Johnson-Lindenstrauss Lemma* by defining the noise matrix as a matrix whose columns have unit lengths [74]. The idea of random projection originated in the following seminal theorem:

  Lemma 2.1. *(Johnson-Lindenstrauss Lemma [75]) Given* $0 < \epsilon < \frac{1}{2}$*, a set* M *of* m *points in* $\mathbb{R}^n$*, upon projection to a uniform random* w*-dimensional subspace where* $w \geq \frac{9 \ln m}{\epsilon^2 - \frac{2}{3}\epsilon^3} + 1$*, the following property holds: with probability at least* $\frac{1}{2}$*, for every pair* $\boldsymbol{x}_1, \boldsymbol{x}_2 \in M$*,*

$$(1-\epsilon)\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 \leq \|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)\|_2^2 \leq (1+\epsilon)\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2. \tag{2.1}$$

  where $f(\boldsymbol{x}_1)$, $f(\boldsymbol{x}_2)$ are the projections of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

  Lemma 2.1 shows that any set of m points in a n-dimensional space can be embedded into an $O(\frac{\ln m}{\epsilon^2})$ dimensional space such that the pairwise distance of any two points is maintained with a high probability (within an arbitrarily small factor). This beautiful property implies that by projecting the data onto a lower dimensional random space through random projection (RP), we can dramatically change its original form while preserving much of its distance-related characteristics.

  From a third party's perspective, these guarantees imply that (a) the users who are close in the original space are likely to remain close in the transformed space and

(b) similarly the users who are far apart are likely to remain so after the transformations. RP matrices can be generated in multiple ways, including:

- Drawing each entry of the matrix independently from the normal distribution $\mathsf{N}(0, 1/w)$ [74,76].
- Drawing each entry of the matrix independently and uniformly at random from $\left\{-\frac{1}{\sqrt{w}}, \frac{1}{\sqrt{w}}\right\}$ [77].
- Drawing each entry of the matrix independently from $\left\{-\sqrt{\frac{3}{w}}, 0, \sqrt{\frac{3}{w}}\right\}$ with probabilities $\left\{\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\right\}$ respectively [77].
- Generating each entry of the matrix by randomised hashing, with the resultant matrix being a sparse matrix [78,79].

Liu *et al.* [80] showed that if the original data set with n attributes is multiplied by a $w \times n$ ($w < n$) mixing matrix, which is random and orthogonal in expectation, then the perturbed data with reduced dimension can be released for learning purpose while preserving much of its distance-related characteristics. However, Sang *et al.* [67] and Liu *et al.* [81] addressed the possible risks of RP by studying how well an attacker can recover the original data from the transformed data with the aid of prior information: if the original data follows a multivariate Gaussian distribution, a large portion of the data can be reconstructed via *maximum a posteriori* (MAP) estimation. Both rotation perturbation and random projection are distance-preserving transformations, which are good for preserving accuracy, but susceptible to attacks that exploit distance relationships [73]. A large portion of the data can be reconstructed via attacks to distance-preserving transformations [70, 71]. Mukherjee *et al.* [82] propose enabling distance-based mining algorithms over private data using Fourier-related transforms, but their approach has the same drawbacks as Liu *et al.* [80]. In order to resist MAP estimation attacks in distance-related applications, we introduce nonlinear transformation as a pre-RP step to condition the pdf of the perturbed data [83].

- *Uniform random transformation* (abbreviated as RT) defines the noise matrix as a matrix whose elements are independently sampled from the same uniform distribu-

tion $U(0,1)$ [24]. As shown in Proposition 4.1, RT does not preserve the distance between data points, this has the advantage of making attacks on distance-preserving transformations [73] not applicable, and it does not distort the distance significantly for low-dimensional data. In particular, RT is used in RMP, where the noise matrix is a projection matrix whose elements are independently sampled from the uniform distribution $U(0,1)$.

Geometric perturbation uses a mix of additive and multiplicative perturbations [84], where the data matrix $\mathbf{X}$ is mapped to $\mathbf{RX} + \mathbf{\Phi} + \mathbf{\Delta}$, where $\mathbf{R}$ is a rotation perturbation matrix, $\mathbf{\Phi}$ is a random translation matrix with identical entries, and $\mathbf{\Delta}$ is an i.i.d. Gaussian noise matrix [72]. It is known that without $\mathbf{\Delta}$, geometric perturbation is vulnerable to "known input attacks" [73], but there are no general results on how the $\mathbf{\Delta}$ term influences the effectiveness of these attacks. All the randomization techniques discussed so far are linear techniques.

Nonlinear transformation is meant to be used in conjunction with linear techniques to thwart Bayesian estimation attacks. The general randomization takes the form $\mathbf{B} + \mathbf{Q} \cdot N(\mathbf{A} + \mathbf{RX})$, where $\mathbf{B}$, $\mathbf{Q}$, $\mathbf{A}$, $\mathbf{R}$ are random matrices, and N is a bounded nonlinear function [85]. The *tanh* function is found to preserve the distance between normal data points, but collapse the distance between outliers, making the function suitable for privacy-preserving anomaly detection [85], provided only the privacy of anomalous records needs to be protected.

The summary of the properties of different randomization techniques are illustrated in Figure 2.3.

To resist both MAP and ICA attacks, *Random Multiparty Perturbation* (RMP) [83] combines RT and nonlinear transformation. The innovations of RMP include (i) the use of participant-specific RT matrices, and (ii) the use of the *double logistic* function as the nonlinear transformation function for protecting both anomalous and normal records from MAP estimation attacks. However, the privacy-preserving properties of the double logistic function have not been thoroughly assessed.

Following RMP, Lyu *et al.* [35] propose an improved two-stage perturbation scheme, which relies on a nonlinear transformation and participant-specific $U(0,1)$-distributed multiplication matrix to resist both MAP and ICA attacks [35]. The improvements in-

Randomization techniques		Vulnerability	Distance-preserving	Solutions
Additive Y= X+R		Spectral filtering attack; PCA attack	✖	R: correlated with X; participant-specific R.
Multiplicative Y= RX	1: Rotation perturbation R: orthogonal	MAP estimation attacks; ICA attacks	✔	Geometric perturbation: depends on Δ Nonlinear perturbation: resists MAP estimation attacks RP: resists ICA attacks
	2: random projection (RP) R : Johnson-Lindenstrauss Lemma	X follows multivariate Gaussian: MAP estimation attacks	✔	
	3: Uniform random transformation (RT) R: uniform	✖	✖	
Nonlinear perturbation Y= N(X)		tanh: normal records suffer from MAP estimation attacks	✖	Improved nonlinear function: Repeated Gompertz (RG)

Figure 2.3: Properties of different randomization techniques.

clude the use of the *repeated Gompertz* function as the nonlinear transformation function for protecting both anomalous and normal records from MAP estimation attacks, and empirical evidence of the advantages of this alternative in terms of the improved recovery resistance of RMP.

In this thesis, Chapter 3 and Chapter 4 realize privacy-preservation through data randomization using the defined metric of recovery rate to asses how much information can be recovered from the perturbed data. To quantify the privacy leakage formally, Chapter 5 and Chapter 6 use differential privacy for privacy-preservation as follows.

2.6.3 Privacy-Preservation through Differential Privacy

Dwork *et al.* [86] first proposed a now standard semantic privacy criterion of *differential privacy* (DP) for the single database scenario, where for every query made, a database server *answers queries* in a privacy-preserving manner with tailored randomization to the response to the query [86]. In comparison with encryption and randomization, differential privacy trades off privacy and accuracy by perturbing the data in a way that is (i) computationally efficient, (ii) does not allow an attacker to recover the original data, and

(iii) does not severely affect the utility. In such schemes, the participants perturb their records or statistics before sending them to the central server or other participants.

Differential privacy provides the theoretical foundation for determining the amount of noise to add to query answers in the statistical disclosure scenario, and does not place limitations on the prior knowledge of the adversary, while ensuring that an attacker with significant prior knowledge and unlimited computational resources cannot determine the presence or absence of a user in the dataset. The concept of differential privacy is that if a tuple is removed from a dataset, the effect on the output likelihood is within a multiplicative factor $1 + \epsilon$.

ϵ-Differential Privacy

For small chosen parameter $\epsilon > 0$, a randomized algorithm $\mathcal{A}$ satisfies ϵ-differential privacy if

$$\Pr\left(\mathcal{A}(D_1) \in \mathcal{S}\right) \leq e^{\epsilon} \Pr\left(\mathcal{A}(D_2) \in \mathcal{S}\right). \tag{2.2}$$

for all measurable $\mathcal{S} \subseteq \text{range}(\mathcal{A})$, and all neighbouring pairs D_1, D_2, where D_1 and D_2 differ in at most one tuple.

Eq. (2.2) can be more easily understood in the following form on probability masses or densities, which is used implicitly in [87]:

$$\left| log \frac{\Pr\left(\mathcal{A}(D_1) = o\right)}{\Pr\left(\mathcal{A}(D_2) = o\right)} \right| \leq \epsilon \tag{2.3}$$

for any output $o \in \text{range}(\mathcal{A})$ with non-zero support, and all neighbouring database D_1, D_2, where D_1 and D_2 differ in at most one record.

In the above equations, the level of indistinguishability is represented by the privacy parameter ϵ, where the lower the value of ϵ, the higher the level of privacy, and more noise is required. ϵ-DP can be attained by adding random noise whose scale is selected through a mechanism that can affect the largest probability of an individual being identified through the query algorithm.

In particular, the level of noise is typically dependent on the sensitivity of the query function, *i.e.*, variation that the removal or substitution of one tuple can cause on the out-

put. In the case of the aggregate count statistic (counting the number of records evaluated to true by any fixed predicte), it is well-known that the sensitivity equals to $\Delta(f) = 1$. However, in many real-world applications, release statistics are real-valued and more complex than a sum, sensitivity must be defined separately. The global sensitivity of a given query indicates the maximum variation of the query output when removing or substituting one element from the database [86].

For answering a recurring query sequence, where the query function f returns a d-dimensional vector ($d > 1$) as the query output, we may use *Vector Sensitivity* of the query function f as the L_2 global sensitivity. Formally, let $f : \mathcal{X}^n \rightarrow \mathbb{R}^d$ be the query function for answering a query sequence with d queries, the *Vector Sensitivity* of f is defined as:

Definition 2.1. *(Vector Sensitivity). For two neighboring datasets D and D′, the L_2 sensitivity of the query function $f : \mathcal{X}^n \rightarrow \mathbb{R}^d$ is expressed as:*

$$\Delta_2 f = \underset{D,D'}{argmax} \, \|f(D) - f(D')\|_2. \tag{2.4}$$

ϵ-differential privacy can be achieved by different mechanisms, among which, Laplace mechanism relies on adding controlled noise to the vector-valued statistics that we want to release. The probability density function of Laplace distribution with μ=0 is: $\text{lap}(x|b)=\frac{1}{2b}e^{-\frac{|x|}{b}}$. Here the variance of Laplace distribution equals $\sigma^2 = 2b^2$. Others, like the exponential mechanism [88] and posterior sampler [89] sample from a problem-dependent family of distributions instead for private maximisation and approximate Bayesian inference, respectively.

Differential privacy has two important composition properties:

- *Sequential Composition*: differential privacy provided by a set of mechanisms M_i each preserving ϵ_i-differential privacy with results on the same database, concatenated, is $\Sigma_i \varepsilon_i$.

- *Parallel Composition*: if every mechanism M_i is executed on disjoint subsets D_i of the private database D, the privacy preserved will be $\max(\varepsilon_i)$-DP.

Theorem 2.1. *(Sequential composition on independent data [90]). Suppose that M^t satisfies ε_t-DP for each $t \in [1, T]$. A combined mechanism $\{M_t, \cdots, M_{t+j}\}$ satisfies $\sum_{k=t}^{k=t+j} \varepsilon_k$-DP.*

(ϵ, δ)-Differential Privacy

(ϵ, δ)-approximate differential privacy [22] relaxes pure ϵ-differential privacy by a *delta* additive term. Unlikely responses need not satisfy the pure differential privacy criterion.

Definition 2.2. *(ϵ, δ)-differential privacy [22]. For scalars $\epsilon > 0$ and $0 \leq \delta < 1$, mechanism $\mathcal{M}$ is said to preserve (approximate) (ϵ, δ)-differential privacy if for all adjacent datasets $D, D' \in \mathcal{D}^n$ and measurable $S \in \text{range}(\mathcal{M})$,*

$$\Pr\left(\mathcal{M}(D) \in S\right) \quad \leq \quad \exp(\epsilon) \cdot \Pr\left(\mathcal{M}(D') \in S\right) + \delta \ .$$

The definition of "adjacent datasets" depends on the application. Most prior work on differentially private machine learning, *e.g.*, [44, 91–94] deals with example-level privacy: two datasets D and D' are defined to be adjacent if D' can be formed by adding or removing a single training example from D.

To achieve (ϵ, δ)-differential privacy, Dwork and Roth describe Gaussian Mechanism [22, Theorem A.1] which adds independent noise scaled to $N(0, \sigma^2)$ to each element of the output, as stated in Theorem 2.2, where $0 < \epsilon < 1$.

Theorem 2.2. *Let $\epsilon \in (0, 1)$ be arbitrary. For $c^2 > 2ln(1.25/\delta)$, the Gaussian mechanism with parameter $\sigma \geq c\Delta_2 f / \epsilon$ is (ϵ, δ)-differentially private, where $\Delta_2 f$ refers to the L_2 sensitivity of query function f. For one-dimensional real-valued functions, the added noise should follow the Gaussian distribution $\mathcal{N}(0, \sigma^2)$. For high-dimensional functions in $\mathbb{R}^d$, the added noise should follow the Gaussian distribution $\mathcal{N}(0, \Sigma)$, where Σ is a diagonal matrix with entries σ.*

To avoid the worst-case scenario of always violating privacy of a δ fraction, the standard recommendation is to choose $\delta \ll 1/N$, where N is the size of the database. This strategy forecloses possibility of one particularly devastating outcome, but other forms of information leakage remain. However, unlike pure differential privacy, there are situations where the approximate differential privacy is not a very elegant abstraction for mathematical analysis, particularly the analysis of composition. There exist valid reasons for preferring the Gaussian mechanism over Laplace: (i) the noise comes from the same Gaussian distribution (closed under convolution) as the error that may already be present in the dataset; (ii) the standard deviation of the noise is proportional to the

query's L_2 sensitivity, which is no larger and often much smaller than L_1 sensitivity; (iii) for the same standard deviation, the tails of the Gaussian distribution decay much faster (more concentrated) than those of the Laplace (exponential) distribution, and so better utility is expected. Another common reason for bringing in (ϵ, δ)-differential privacy is application of the advanced composition theorem, as shown in Theorem 2.4, which provides a stronger privacy guarantee compared with the basic composition theorem in Theorem 2.3. Consider the case of k-fold adaptive composition of an (ϵ, δ)-DP mechanism. For any $\delta' > 0$, it holds that the composite mechanism is $(\epsilon', k\delta + \delta')$-DP, where $\epsilon' = \sqrt{2kln(1/\delta')\epsilon} + k\epsilon(e^{\epsilon} - 1)$ [95].

Theorem 2.3. *Basic composition theorem of (ϵ, δ)-differential privacy (the epsilons and the deltas add up): the composition of m differentially private mechanisms is $(\sum_i \epsilon_i, \sum_i \delta_i)$-differentially private, where for any $1 \leq i \leq m$, the ith mechanism M_i is (ϵ_i, δ_i)-differentially private.*

Theorem 2.4. *Advanced composition theorem of (ϵ, δ)-differential privacy [95] (subsequently improved by [96,97]): the composition of k mechanisms which are each (ϵ, δ)-differentially private is $(\sqrt{k}\epsilon, k\delta)$-differentially private.*

A key property of DP is the post-processing guarantee [22]: any data-independent computation applied on the output of an (ϵ, δ)-DP randomized algorithm remains (ϵ, δ)-DP.

The most recent progress in DP uses the privacy definitions of *Concentrated Differential Privacy* (CDP) by Dwork and Rothblum [98], followed by *zero-Concentrated Differential Privacy* (zCDP) by Bun and Steinke [99]. They are framed using the language of, respectively, subgaussian tails and the Rényi divergence. *Rényi Differential Privacy* (RDP) is recently introduced by Mironov [100]. RDP is closely related to *(zero)-Concentrated Differential Privacy* [98, 99]. Both Concentrated and zero-Concentrated DP require a linear bound on all positive moments of a privacy loss variable. By contrast, Rényi DP applies to one moment at a time, thus facilitating tracking privacy budget across multiple queries. Although less restrictive, it allows for more accurate numerical analysis.

Distributed Differential Privacy (DDP)

Differential privacy was designed for the scenario where a trusted database server, which is entitled to see all participants' data in the clear, wishes to *publish statistics or answer queries* in a privacy-preserving manner by randomizing query results [86]. In such a scenario, the database comprises private data of *multiple individuals*. However, for the scenario where data are sourced from multiple sensitive databases, while the aggregator is untrusted or no aggregator exists, for example, in a participatory sensing scenario, participants are data owners who *publish data* (instead of answering queries) about *themselves alone*, a distributed version of differential privacy—*Distributed Differential Privacy* (DDP)—is needed [10,50]. The notion of DDP reflects the fact that the noise in the target statistic is sourced from multiple parties.

In the same year differential privacy was proposed, Dwork *et al.* [50] proposed the protocol called "Our Data, Ourselves" (ODO) to achieve differential privacy through distributed randomness. However, the main challenge of ODO is the cooperative generation of noise, which incurred a cost of $O(n)$ multiplications and additions in shares. Furthermore, it requires pairwise secure channels among parties, which is incompatible with scenarios where parties do not realistically share pairwise connections with each other, such as smart grids.

Shi *et al.* [10] first defined DDP for neighboring vectors that differ in exactly one coordinate; we reformulate DDP for decentralized aggregation in Chapter 5.

For aggregation tasks, to ensure differential privacy for the target statistic (aggregation), the target statistic must contain random noise of an appropriate magnitude. One naive solution is to rely on a single participant to add the required magnitude of noise. However, this solution is problematic, because this designated participant knows the noise and hence can deduce from the output the true aggregate value. In real-world settings, participants may not trust each other. In particular, a subset of the participants may be compromised and collude with the data aggregator. In the worst case, for a total of N participants, if every participant believes that the other $N-1$ participants may be compromised and collude with the aggregator, each participant would need to add sufficient noise to ensure local differential privacy (LDP). The resulting statistic would accumulate significant error. If at least a fraction of the participants are honest and not compromised,

then we can distribute the noise generation task amongst these honest participants.

Theorem 2.5. *(Cramér's decomposition theorem [101]) If X and Y are independent real-valued random variables whose sum $X + Y$ is a normal random variable, then both X and Y must be normal as well.*

One way to realize DDP is the use of a stable Gaussian distribution to perturb individual statistics. By induction from Cramér's decomposition in Theorem 2.5, for any finite sum of independent real-valued random normal variables, their summands must also be normal. This nice decomposition property of Gaussian distribution facilitates the distribution of noise generation among all parties, who jointly generate the required noise r with standard deviation σ in the aggregate statistic, thus each party may add less noise ($\sigma_i = \frac{\sigma}{\sqrt{n}}$), and as long as the noise in the desired statistic (aggregation/sum/average) has an expected standard deviation of σ, privacy can be guaranteed.

However, since DDP collects noise from all parties, only the sum of the individually released statistics is differentially private but not the individually shared statistics, *i.e.*, $\sum r_i$ is sufficient for differential privacy, but r_i alone is not sufficient, thus $x_i + r_i$ cannot be released directly, here x_i indicates the plaintext. Therefore, this necessitates the help of cryptographic techniques to maintain utility and ensure aggregator obliviousness, as evidenced in [8–10, 102]. Different schemes for differentially private aggregation are outlined in Table 2.1.

Table 2.1: Differences between differentially private aggregation schemes.

Schemes	Untrusted aggregator	Interaction	Crypto	DDP	Transformation	Support failure
Rastogi *et al.* [9]	Yes	Yes	Yes	Yes	DFT	Yes
Shi *et al.* [10]	Yes	Yes	Yes	Yes	No	No
Ács *et al.* [8]	Yes	No	Yes	Yes	No	Yes
Lyu *et al.* [102]	Yes	No	Yes	Yes	No	Yes

In distributed scenario, without the help of cryptographic techniques, each participant has to add enough calibrated noise to ensure LDP, the aggregated noise leads to huge utility degradation. This inspires us to explore DDP using stable distributions to ensure differential privacy of the target statistic, while preserving individual privacy by combing DDP with cryptosystem in Chapter 5 and Chapter 6.

2.7 Summary

Most of the current machine learning frameworks require a central server to build a global model on all participants' data, aggregate local statistics, or mediate the collaborative learning process. While the benefits brought by server-based frameworks are encouraging in various applications, there exists some inherent limitations. Decentralized machine learning aims to overcome these limitations by parallelizing computation among participants. On the other hand, fog-empowered data aggregation is well positioned for real-time large-scale IoT network. However, there are increasing concerns about the privacy leakage of sensitive information in both machine learning and data aggregation. To address these concerns, privacy-preserving techniques have enabled to extract useful knowledge from the combined data or local statistics without compromising utility. This chapter presents an overview of different machine learning frameworks, adversary models, privacy attacks, and systematically investigates various privacy-preserving techniques, including encryption, data randomization, and differential privacy. It has noted that the main objective of privacy-preserving machine learning and data aggregation are two fold: preserve privacy and maintain utility.

Part I

Centralized and Decentralized Privacy-Preserving Machine Learning

The journey of a thousand miles begins with one step.

Lao Tzu

3

Privacy-Preserving Collaborative Anomaly Detection

With regards to real-world applications, our first example of privacy-preserving machine learning is collaborative anomaly detection. Anomaly detection is important for tasks such as security, fault diagnosis, intrusion detection, and monitoring applications. Given the possibility that a cloud service is honest but curious, a major challenge is to protect the participants' privacy when collaborating via a cloud server. In order to identify anomalies in the combined data from multiple participants, it is necessary to build an anomaly detection model on the server. However, considering privacy issues of forwarding raw sensitive data to the server, Erfani *et al.* [83] proposed a privacy-preserving collaborative anomaly detection scheme called *Random Multiparty Perturbation* (RMP) that allows participants to perturb their tabular data by passing the data through a nonlinear function, and projecting the perturbed data to a lower dimension using a participant-specific random matrix. The perturbation process of RMP consists of a nonlinear transformation

stage and a linear projection stage. The nonlinear stage is used to condition the *probability density function* (pdf) of the perturbed data to thwart *maximum a posteriori* (MAP) estimation attacks, whereas the linear stage is to compress the data to resist *independent component analysis* (ICA) attacks. The nonlinear transformation function of RMP is the *double logistic* function, but the privacy-preserving properties of this function have not been thoroughly assessed. As such, we propose an improvement to RMP by introducing a new nonlinear function. The improved scheme is assessed in terms of its recovery resistance to MAP estimation attack, and it is shown to deliver better recovery resistance and an improved trade-off between accuracy and privacy.

3.1 The Improved RMP

We propose a nonlinear transformation function called "repeated Gompertz", and experimentally demonstrate that the proposed repeated Gompertz function is more resistant to MAP estimation attacks than the double logistic function proposed in [83]. The general participatory sensing architecture is depicted in Figure 3.1. It comprises a set of participants $\mathcal{C} = \{c_i | i = 1, \ldots, q\}$, a cloud service $\mathcal{S}$, and a set of end-users $\mathcal{U}$. The cloud service is assumed to be honest but curious, *i.e.*, it will never perform any malicious action to disrupt the protocols or compromise the participants but it might try to discover privacy-sensitive information of the participants, including colluding with some of the participants. RMP here considers the case where (i) the main learning task is anomaly detection, and (ii) scalability requirement necessitates the use of a randomisation-based scheme. Based on the state-of-the-art in privacy-preserving research, the following design criteria are considered:

- **Resilience to distance inference attacks**: Uniform random transformation [24] does not preserve the angle (inner product) or Euclidean distance between transformed data points, and is thus resistant to distance inference attacks. Moreover, it is suitable for anomaly detection [83].

- **Resilience to Bayesian estimation attacks**: Bayesian estimation is a general attack that exploits the pdf of the original data. Gaussian data is particularly ex-

ploitable because it reduces the MAP estimation problem to a simple convex optimisation [67]. A nonlinear transformation can be applied to condition the pdf.

- **Resilience to collusion**: Let $\mathbf{X}_i \in \mathbb{R}^{n \times m_i}$ be the dataset of participant c_i, where m_i and n denote the number of records and features respectively. If participant c_i perturbs its records as $\mathbf{Z}_i = \mathbf{T}\mathbf{X}_i$, where $\mathbf{T} \in \mathbb{R}^{w \times n}$, $w < n$, is a random matrix shared by all participants, then leakage of $\mathbf{T}$ due to collusion with the cloud service S can compromise the privacy of all the other participants. Preventing collusion requires each participant to use an independent perturbation matrix.

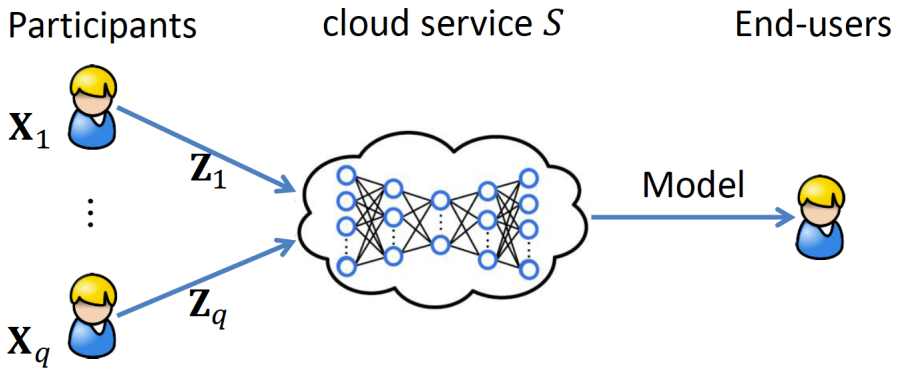


Figure 3.1: The general participatory sensing architecture.

RMP's two-stage data perturbation scheme was designed with the preceding criteria in mind. Let $\mathbf{T}$ be a $w \times n$ matrix ($w < n$) with $U(0,1)$-distributed elements. Each participant c_i generates a unique perturbation matrix

$$\tilde{\mathbf{T}}_i = \mathbf{T} + \Delta_i, \tag{3.1}$$

where each element of Δ_i is drawn from $U(-\alpha, \alpha)$, and $0 < \alpha < 1$, which indicates the level of imposed noise in RMP anomaly detection. Experimental results show that for small values of α, the accuracy loss in anomaly detection is small [83]. Suppose participant c_i is contributing data $\mathbf{X}_i \in \mathbb{R}^{n \times m_i}$ to the cloud service S for anomaly detection. The participant transforms $\mathbf{X}_i$ to $\mathbf{Z}_i \in \mathbb{R}^{w \times m_i}$ in two stages:

Stage 1: Participant c_i transforms $\mathbf{X}_i$ to $\mathbf{Y}_i$, by applying the nonlinear perturbation function N element-wise:

$$\mathbf{Y}_i = N(\mathbf{X}_i). \tag{3.2}$$

In the original version of RMP, N is defined as the double logistic function. Here for the improved RMP, N is chosen to be the repeated Gompertz function:

$$\begin{aligned} N(x) \overset{\text{def}}{=} & a_1 e^{-b_1 e^{-c_1 x - d_1}} u(0.35 - x) \\ & + \left(0.5 + a_2 e^{-b_2 e^{-c_2 x - d_2}}\right) u(x - 0.35), \end{aligned} \tag{3.3}$$

where the parameters $a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2$ are defined in Figure 3.2, and $u()$ is the Heaviside step function. The derivation of the function parameters is explained in Sect. 3.2. Figure 3.2 plots different nonlinear perturbation functions for comparison.

Stage 2: Using $\tilde{\mathbf{T}}_i$ generated earlier in Eq. 3.1, the participant transforms $\mathbf{Y}_i$ to $\mathbf{Z}_i$:

$$\mathbf{Z}_i = \tilde{\mathbf{T}}_i \mathbf{Y}_i. \tag{3.4}$$

The participant then sends $\mathbf{Z}_i$ to the cloud service $\mathcal{S}$. Once $\mathcal{S}$ receives all the perturbed datasets $\mathbf{Z}_i$, $i = 1, \ldots, q$, it concatenates them as: $\mathbf{Z}_{\text{all}} = [\mathbf{Z}_1 | \cdots | \mathbf{Z}_q]$, and then trains an anomaly detection model on $\mathbf{Z}_{\text{all}}$. The learned model $\mathcal{M}$ can be used by end-users to identify anomalies in their test records. RMP is independent of the anomaly detection algorithm used: denoising autoencoder (DAE) is used for anomaly detection, forward reference to Sec 3.3.2 for more details.

3.2 Privacy Analysis

3.2.1 Maximum a Posteriori (MAP) Estimation Attack

Corresponding to multiplicative perturbation, we use a specific privacy-preserving definition: a perturbation scheme is privacy-preserving with respect to an attack $\mathcal{A}$ and a data distribution p_D if only a small fraction of the original data, characterised by p_D, can be recovered from the perturbed data through $\mathcal{A}$. This definition has three major components:

- the *reference attack*;
- the *data distribution*, which captures an aspect of the attacker's auxiliary information;

Param	Value
a_1	0.5247
b_1	6.0000
c_1	13.2000
d_1	0.2040
a_2	0.4000
b_2	6.0000
c_2	27.5000
d_2	-16.0750

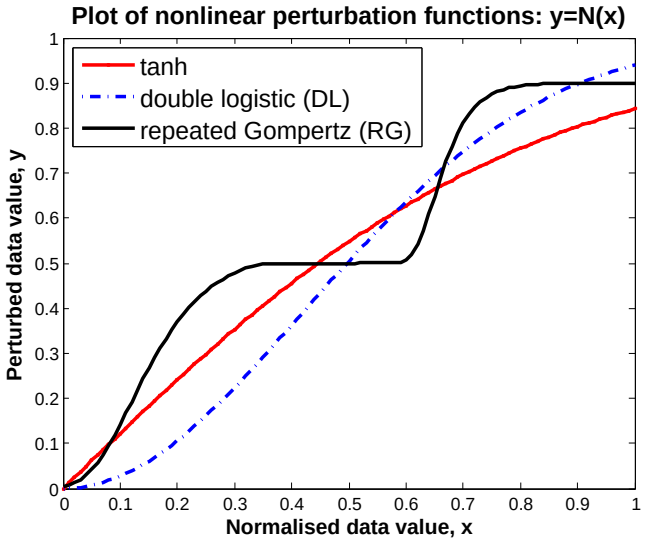


Figure 3.2: Parameters of the repeated Gompertz function, and a plot of different nonlinear perturbation functions. The tanh function is $N(x) = \tanh(\beta_t x)$, where $\beta_t \approx 1.23$ [83]. The double logistic function is $N(x) = 1 - \exp(-\beta_{dl} x^2)$, where $\beta_{dl} \approx 2.81$ [83]. The repeated Gompertz function is defined as per Eq. (3.3).

- the *recovery rate*, which captures the notions of "small fraction" and "recovered".

To specify the reference attack, we first consider attacks to linear multiplicative perturbation schemes. These types of schemes project the whole data matrix to a lower dimensional space so that an attacker has only an ill-posed problem in the form of an underdetermined system of linear equations $\mathbf{T}\boldsymbol{y} = \boldsymbol{z}$ to work with, where $\boldsymbol{z}$ is a projection of vector $\boldsymbol{y}$. An underdetermined system cannot be solved for $\boldsymbol{y}$ exactly, but given sufficient prior information about $\boldsymbol{y}$, an approximation of the true $\boldsymbol{y}$ may be attainable. Below, we characterise an attack by the extent of prior information available to the attacker.

In a *known input-output attack*, the attacker has some input samples and all samples of the perturbed data, and knows which input sample corresponds to which output sample [73]. In the participatory sensing scenario where the cloud service may collude with one or more participants to unravel other participants' data, the known input-output attack is an immediate concern. In the following, our privacy analysis is conducted with respect to a known input-output attack based on MAP estimation — this is our reference attack. MAP estimation is based on Bayesian statistics and is more general than maximum likelihood estimation because the former takes an arbitrary prior distribution into account.

To measure the strength of the reference attack, we define the recovery rate. If for a data vector $\boldsymbol{x}$ the recovered copy is $\hat{\boldsymbol{x}}$, then the *relative error* is $\xi \stackrel{\text{def}}{=} \|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2 / \|\boldsymbol{x}\|_2$, where $\|\cdot\|_2$ is the Euclidean norm. Denote the joint distribution of ξ and $\boldsymbol{x}$ by $p_{\Xi,\boldsymbol{X}}(\xi, \boldsymbol{x})$, then we define the *$\epsilon$-recovery rate* with respect to the perturbation algorithm and attack as

$$r_\epsilon(\mathcal{A}, p_D) \stackrel{\text{def}}{=} \int_{\xi=0}^{\epsilon} \int_{\boldsymbol{x} \in D_{\boldsymbol{x}}} p_{\Xi,\boldsymbol{X}}(\xi, \boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\xi, \tag{3.5}$$

where $D_{\boldsymbol{x}}$ is the domain of the data vector, and $\boldsymbol{x}$ is normalised. The joint distribution $p_{\Xi,\boldsymbol{X}}$ depends on the attack $\mathcal{A}$ and data distribution p_D. In the absence of an analytical expression for Eq. (3.5), we estimate the recovery rate as the fraction of test data that can be recovered to within a relative error of ϵ. At this point, we state the privacy definition formally as follows.

Definition 3.1. *A probabilistic algorithm that takes p_D-distributed $\boldsymbol{x} \in \mathbb{R}^n$ as input and produces $\boldsymbol{z} \in \mathbb{R}^w$ as output is* (ϵ, δ)-recovery resistant *with respect to p_D and attack algorithm $\mathcal{A}$ if $r_\epsilon(\mathcal{A}, p_D) = \delta$.*

Suppose the attacker is targeting a particular participant by trying to solve $\mathbf{Z} = \mathbf{T}\mathbf{Y}$ for $\mathbf{Y}$. We consider two scenarios: where $\mathbf{T}$ is known, and where $\mathbf{T}$ is unknown.

- *Scenario where random matrix $\mathbf{T}$ is known.* This is the worst-case scenario and here, we assume the attacker somehow knows $\mathbf{T}$ exactly but not $\mathbf{Y}$, for example when the attacker manages to predict the output of the victim's improperly initialised pseudorandom number generator (in fact, such a vulnerability was discovered on the Android mobile platform in mid-2013). Let $\boldsymbol{z}$ represent a column of $\mathbf{Z}$, and $\boldsymbol{y}$ represent a column of $\mathbf{Y}$. The MAP estimate of $\boldsymbol{y}$, given $\mathbf{T}$ and $\boldsymbol{z}$, is

$$\begin{aligned} \hat{\boldsymbol{y}} &\in \arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{z}, \mathbf{T}) = \arg\max_{\boldsymbol{y}} \frac{p(\boldsymbol{z}|\mathbf{T}, \boldsymbol{y}) p(\mathbf{T}) p(\boldsymbol{y})}{p(\boldsymbol{z}|\mathbf{T}) p(\mathbf{T})} \\ &= \arg\max_{\dagger \in \mathcal{Y}} \frac{p(\boldsymbol{y})}{\int_{\mathbb{R}^n} p(\boldsymbol{z}|\mathbf{T}, \boldsymbol{y}) \, \mathrm{d}\boldsymbol{y}} = \arg\max_{\dagger \in \mathcal{Y}} p(\boldsymbol{y}), \end{aligned} \tag{3.6}$$

  where $\mathcal{Y} = \{\boldsymbol{y} : \boldsymbol{z} = \mathbf{T}\boldsymbol{y}\}$.

  - The factor $p(\boldsymbol{z}|\mathbf{T}, \boldsymbol{y})$ translates to the constraint $\boldsymbol{y} \in \mathcal{Y}$.
  - The integral in the denominator does not contribute towards maximising $\boldsymbol{y}$.

If $\boldsymbol{y}$ is n-variate Gaussian with a positive definite covariance matrix, then Eq. (3.6) becomes an easily solvable quadratic programming problem [67, Theorem 1]. The key is to design a nonlinear function N that transforms a potentially Gaussian data distribution to a distribution that deters accurate solution of Eq. (3.6).

For nonlinear function, Bhaduri *et al.* [85] proposed tanh, *i.e.*, $N(x) = \tanh(\beta_t x)$, where β_t is a tunable parameter. As a suitable value of β_t, we pick

$$\beta_t = \arg\min_{\beta} \int_0^1 [\tanh(\beta x) - x]^2 \, \mathrm{d}x \approx 1.23.$$

Erfani *et al.* [83] proposed using the double logistic function, *i.e.*, $N(x) = \text{sign}(x)[1 - \exp(-\beta_{dl} x^2)]$, where

$$\beta_{dl} = \arg\min_{\beta} \int_0^1 [1 - \exp(-\beta x^2) - x]^2 \, \mathrm{d}x \approx 2.81.$$

We propose "repeated Gompertz" function defined in Eq. (3.3) as the nonlinear function. The design principles of the proposed function are explained as follows. The Gompertz function takes the standard form:

$$\text{Gompertz}(x) = ae^{-be^{-cx}}, \tag{3.7}$$

where the parameter a specifies the upper asymptote, b controls the displacement along the x axis, and c adjusts the growth rate of the function. As $\tanh(\beta_t x)$ is good for protecting anomalous data points, the repeated Gompertz function is given slopes that approximate those of $\tanh(\beta_t x)$ at $x = 0$ and $x = 1$. In order to protect normal data points, the repeated Gompertz function is also designed to have a flat middle section so that for that section the function cannot be inverted. Through extensive search, we found the geometry in Figure 3.2 to be good for protecting both anomalous and normal data points: (i) a Gompertz curve presenting a steep slope over the interval $[0, 0.35]$; and (ii) another Gompertz curve presenting a plateau over the interval $[0.35, 0.6]$, a steeper slope over the interval $[0.6, 0.75]$ and another plateau over the interval $[0.75, 1]$. The parameters of the two Gompertz functions are given in Figure 3.2. This compositional structure inspired the name "repeated

Gompertz".

Both tanh and double logistic function are readily invertible, while repeated Gompertz is not, thus making an analytical expression of the transformed pdf unavailable even if the pdf of the original random variable is available. It means there does not exist a distribution of x for which the attacker can find an analytical expression for p(y) in Eq.(3.6), where y is the result of applying repeated Gompertz to x.

- *Scenario where random matrix* $\mathbf{T}$ *is unknown.* Consider the case where the attacker knows neither $\mathbf{T}$ nor $\mathbf{Y}$. The MAP estimates of $\mathbf{Y}$ and $\mathbf{T}$, given $\mathbf{Z}$, are

$$\begin{aligned}(\hat{\mathbf{T}}, \hat{\mathbf{Y}}) &\in \arg\max_{\mathbf{T},\mathbf{Y}} p(\mathbf{T}, \mathbf{Y}|\mathbf{Z}) \\ &= \arg\max_{\mathbf{T},\mathbf{Y}} \frac{p(\mathbf{Z}|\mathbf{T}, \mathbf{Y})p(\mathbf{T})p(\mathbf{Y})}{\int\int p(\mathbf{Z}|\mathbf{T}, \mathbf{Y})p(\mathbf{T})p(\mathbf{Y})d\mathbf{T}d\mathbf{Y}} \\ &= \arg\max_{(\mathbf{T},\mathbf{Y})\in\Theta} p(\mathbf{T})p(\mathbf{Y}),\end{aligned} \tag{3.8}$$

where $\Theta \in \{(\mathbf{T}, \mathbf{Y}) : \mathbf{Z} = \mathbf{TY}\}$. In a known input-output attack, $p(\mathbf{T})$ and $p(\mathbf{Y})$ are estimated as inputs to Eq. (3.8). Eq. (3.8) is a nonconvex optimisation problem that is harder to solve than Eq. (3.6). The repeated Gompertz is designed to make data recovery via Eq. (3.6) difficult when $\mathbf{T}$ is known. Now that $\mathbf{T}$ is unknown, the attacker is expected to get an even lower recovery rate by solving Eq. (3.8), which is a more difficult problem.

3.2.2 Underdetermined Independent Component Analysis (UICA) Attack

As a statistical technique, ICA represents a set of random variables as linear combinations of statistically independent component variables. The aim of an ICA attack is to design a filter that can recover the original signals from only the observed mixture. ICA can separate out $\mathbf{T}$ and $\mathbf{Y}$, knowing only their product $\mathbf{Z} = \mathbf{TY}$, provided (i) The number of observed attributes is at least as large as the independent attributes, $w \geq n$; (ii) the attributes are independent; (iii) at most one of the attributes is Gaussian; (iv) $\mathbf{T}$ must have full column rank. To resist an ICA attack, we enforce $w < n$, namely projecting data to a lower-dimensional subspace to make the problem of ICA underdetermined. In this

case, even if the perturbation matrix **T** is known, the independent components cannot be obtained. Moreover, as shown in [80], if $w \leq (n+1)/2$, no linear filter can separate out the observed mixture **Z**. It is demonstrated that an ICA attack cannot effectively breach the privacy of participants after random projection-based perturbation.

3.3 Simulations and Evaluation

This section presents the simulation and evaluation results of the improved RMP in terms of its privacy-preserving and accuracy-preserving properties. Since it is resistant to ICA, the empirical experiments focus on the recovery rate of our scheme under the MAP estimation attack and compare with earlier works, *i.e.*, [80], [85] and [83].

Experiments are conducted on (i) purely Gaussian datasets, (ii) purely Laplace-distributed datasets, (iii) seven real datasets from the UCI Machine Learning Repository[1], and (iv) two challenge synthetic datasets. Seven real datasets are (i) Abalone, (ii) Forest, (iii) Adult, (iv) Gas, (v) OAR, (vi) DSA, and (vii) HAR, with dimensionalities of 8, 54, 123, 128, 110, 315 and 561 respectively. The two synthetic datasets are (i) Smiley with 20 features, and (ii) GME with 100 features. The Smiley dataset consists of samples drawn from two compact Gaussians and an arc shaped distribution to resemble a smiley face, and is often used to challenge anomaly detection algorithms. The GME dataset is a mixture of four separated Gaussians.

Among these datasets, HAR dataset [103] belongs to wearable data. The experiment is carried out with a group of 30 volunteers. Each person performes six activities (WALKING, WALKING UPSTAIRS, WALKING DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist.

3.3.1 Privacy Evaluation

Experimental results are provided in this section on the recovery resistance of the improved RMP against the MAP estimation attack, in terms of the ϵ-recovery rate defined in Eq. (3.5). In the absence of an analytical expression for Eq. (3.5), we estimate the ϵ-recovery rate as the fraction of test data that can be recovered to within a relative error of

[1] https://archive.ics.uci.edu/ml/index.php

ϵ (see Definition 3.1):

$$\hat{r}_\epsilon(\mathcal{A}, p_D) \stackrel{\text{def}}{=} \frac{\#\left\{\hat{\boldsymbol{x}}_i : \frac{\|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|_2}{\|\boldsymbol{x}_i\|_2} \leq \epsilon, i = 1, \ldots, m\right\}}{m}, \tag{3.9}$$

where $\boldsymbol{x}_i$ and $\hat{\boldsymbol{x}}_i$ are the ith original data record and its attacker-estimated value respectively.

To execute MAP estimation, the attacker can either apply the [67, Theorem 1] formula, provided the original data is multivariate Gaussian distributed; or solve the constrained optimisation problem (3.6). To solve optimisation problem (3.6), the attacker needs to evaluate an objective function that is the pdf of the original data. For this, the attacker can estimate the pdf of the original data as the pdf of the leaked input samples, using multivariate *kernel density estimation* (KDE). For KDE, we use Ihler and Mandel's Kernel Density Estimation Toolbox for MATLAB[2]. Among the kernels supported, we use the Epanechnikov kernel — which is optimal in the sense of the asymptotic mean integrated squared error — with uniform weights.

Table 3.1: Evaluated schemes.

Scheme	Nonlinear perturbation function (stage 1)	Linear projection matrix (stage 2)
RP [80]	none	$\mathbf{T} \sim N_{w\times n}(0, 4)$
tanh+RT	tanh [85]	$\mathbf{T} \sim U_{w\times n}(0, 1)$
DL+RT [83]	double logistic	$\mathbf{T} \sim U_{w\times n}(0, 1)$
RG+RT	repeated Gompertz	$\mathbf{T} \sim U_{w\times n}(0, 1)$

The four schemes shown in Table 3.1 are evaluated in the *worst-case* scenario where the attacker knows exactly the victim's perturbation matrix.

Purely Gaussian datasets: Figure 3.3 shows that RG+RT provides significantly higher recovery resistance for both normal and anomalous data compared to the other schemes, except 0.2-recovery rate for anomalous data case, which is slightly less effective than RP.

[2] http://www.ics.uci.edu/~ihler/code/kde.html

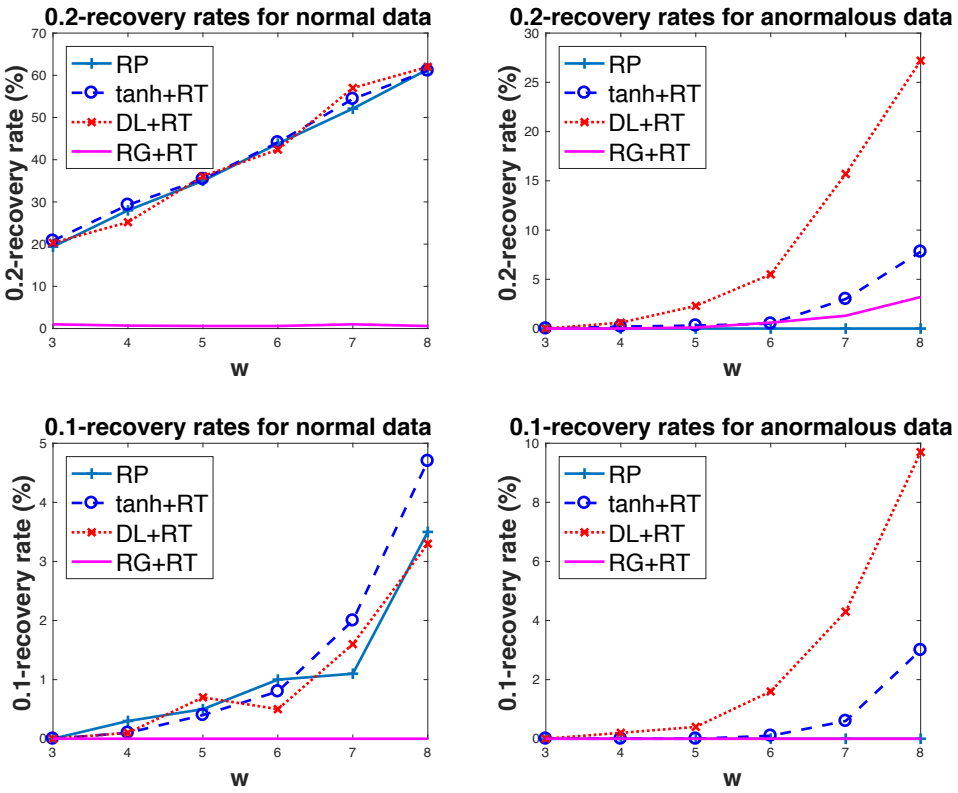


Figure 3.3: Recovery rates of MAP estimation attacks against the evaluated schemes, on $w \times 1000$ data projected from 15×1000 normalised Gaussian-distributed data (zero mean, identity covariance matrix).

Purely Laplace datasets: Figure 3.4 shows that RG+RT significantly outperforms other methods for Laplace datasets, and this is especially evident for normal data. In particular, for both normal data and anormalous data, the 0.2-recovery rate and 0.1-recovery rate against RG+RT are below 10%. It should be noted that for anormalous data, the 0.2-recovery rate and 0.1-recovery rate against tanh+RT are also below 10%, but still RG+RT outperforms tanh + RT.

Assorted real and synthetic datasets: Consistent with the results for purely Gaussian and purely Laplace datasets, as shown in Figure 3.5, RG+RT also outperforms tanh+RT and DL+RT in terms of recovery resistance for both normal data and anomalous data. Note the low recovery rates in many cases, especially for example, RG+RT achieves (0.1, 0)-recovery resistance for the DSA and HAR datasets. It should be noted that for DSA and HAR datasets, all the other schemes also show negligible values.

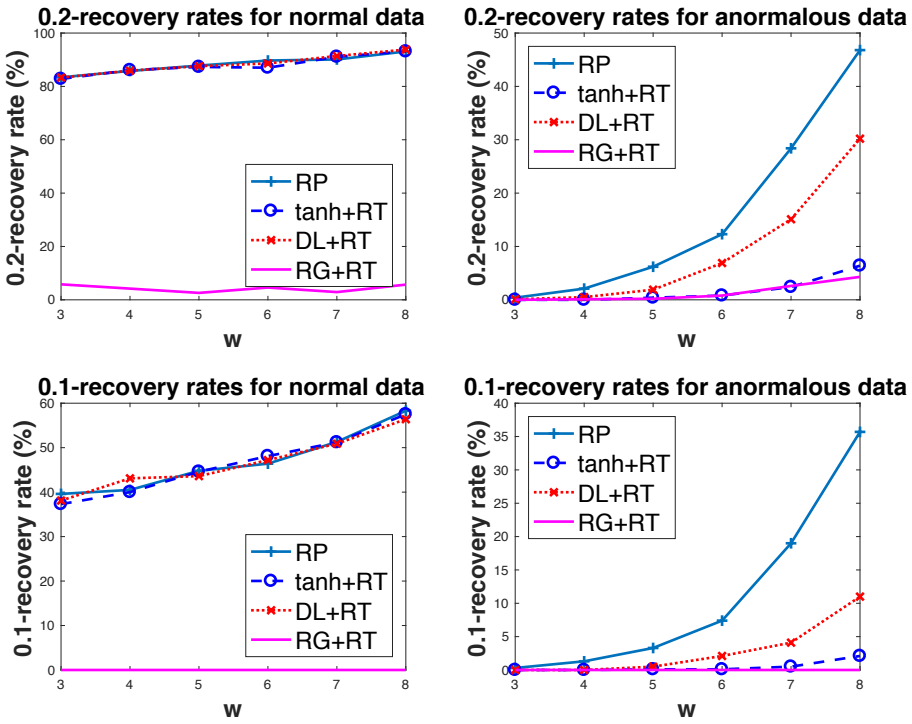


Figure 3.4: Recovery rates of MAP estimation attacks against the evaluated schemes, on $w \times 1000$ data projected from 15×1000 normalised Laplace-distributed data (zero mean, unity scale).

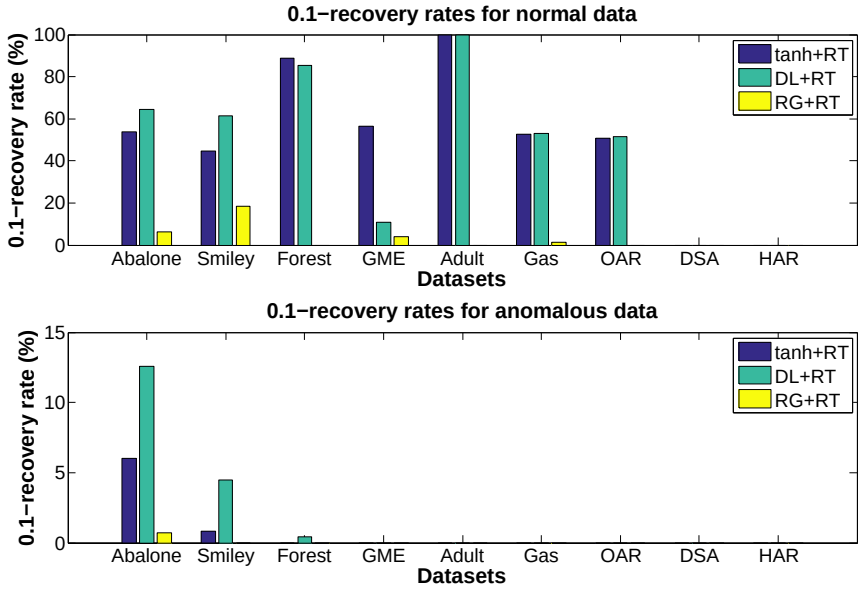
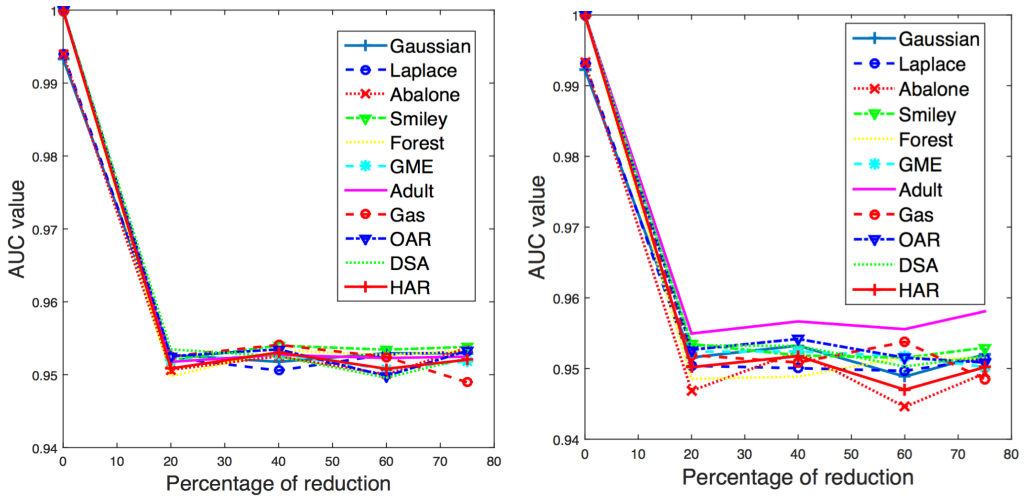


Figure 3.5: 0.1-recovery rates of the MAP estimation attack against the evaluated schemes, on various datasets. The rank of the perturbation matrix, w, is set as $\lfloor (n+1)/2 \rfloor$, where n is the number of features. Note zero recovery rates in many cases.

3.3.2 Accuracy Evaluation

The Area Under the ROC Curve (AUC) is used to compare the anomaly detection accuracy with and without data perturbation by our scheme. Without data perturbation, the AUC should be close to 1. With data perturbation, the AUC is expected to decrease, and the goal is to measure the extent of this decrement. Reducing data dimensionality from n to $w \leq (n+1)/2$ ensures that no *linear* filter can recover the original data from its perturbed version [80]. On the other hand, this raises the concern of utility loss.

For anomaly detection, a stacked *denoising autoencoder* (DAE) is used [104], the hyperparameters of the DAE are set based on the best performance on validation set. Each dataset is normalised to $[0,1]$ and merged with 5% anomalous records, which are distributed between $[0, 0.05]$ or $[0.95, 1]$. Anomalies are identified by DAE based on the mean absolute error between the inputs and outputs of the training records. According to three sigma rule, a well-known measure for anomaly detection, the reconstruction error is expected to be Gaussian distributed, hence 99.73% of the error values are expected to be at most three standard deviations away from the mean, namely, within the threshold $\mu(e) + 3\sigma(e)$. An error value larger than the threshold is unlikely and is identified as an anomaly.



(a) T changes (b) T,N change

Figure 3.6: Impact of RG-RP scheme on AUC with T changes, and with T,N change. The x-axis shows the percent reduction in the number of dimensions, $(n-w)/n \times 100\%$.

To study the performance of our method, two experiments are conducted. In the first

experiment, the nonlinear function is set to $N = RG$ with the fixed coefficients in Figure 3.2, while Δ_i of each participant c_i is randomly drawn from $U(-0.1, 0.1)$. Figure 3.6(a) shows the results for the first experiment, our two-stage scheme with changing $\mathbf{T}$ only has minor impact on accuracy. During the second stage, reducing data dimensionality by 50% decreases the detection rate by at most 5% in the worst case. Our study also reveals that RG+RT has better or similar AUC performance compared with tanh+RT and DL+RT.

In the second experiment, in addition to varying the transformation matrix $\mathbf{T}$, the coefficients of the nonlinear function are also randomly modified for each participant. The second experiment explores the impact of randomising both $\mathbf{T}$ and N, as opposed to only randomising $\mathbf{T}$. The coefficients are perturbed with some random noise, for example, a_i is perturbed as $a_i + \Delta$, for $i = 1, 2$, where Δ is randomly drawn from a Gaussian distribution $N(0, 1)$. While randomising the coefficients of the nonlinear transformation N might be desirable, as it preserves privacy better, it is interesting to assess its impact on accuracy. Figure 3.6(b) shows that randomising the coefficients of N has little impact on the accuracy of DAE, *i.e.*, reducing the accuracy not more than 1% compared to using a fixed N as in Figure 3.6(a).

3.4 Summary

For cloud server-based privacy-preserving anomaly detection, we propose a two-stage perturbation scheme, which uses a new nonlinear transformation function called "repeated Gompertz". Our scheme maintains the privacy of both normal and anomalous records in terms of resistance to both MAP estimation attacks and ICA attacks. We use *ϵ-recovery rate* to evaluate the effectiveness of our scheme, it measures how much data can be recovered within a relative error of ϵ. Extensive experiments on both real-world and synthetic datasets demonstrate that our proposed repeated Gompertz function is more resistant to MAP estimation attacks than the state-of-the-art nonlinear functions.

When angry, count to four; when very angry, swear.

Mark Twain

4

Privacy-Preserving Collaborative Human Activity Recognition

In ubiquitous computing, wearable *Human Activity Recognition* (HAR) is attracting significant interest. HAR aims to recognise human activities via inertial, video or other sensors, in support of ambient intelligence in smart environments, in which people with care needs can be monitored and timely cared for, for example. Furthermore, with the tremendous growth of data not only in volume, but also in the number of features, we are motivated to investigate how to apply our proposed privacy-preserving schemes to large-scale machine learning, such as human activity recognition. As a participant-specific perturbation matrix breaks the relationship between data points by adding an additional participant-specific noise to a uniform distributed matrix, this potentially limits its application to distance-based data analyses. Hence, random transformation (*e.g.*, RT of Chapter 3) is replaced with random projection (RP) in the second stage for more accurate recognition.

Movement information gathered from body-worn sensors are multivariate time-series data with inherent local dependency characteristics and relatively high spatial and temporal resolution. For analyzing this kind of data, *hidden Markov models* (HMMs) have been widely used, owing to their capability for temporal pattern decoding. An HMM assigns probability values over unbounded sequences. Since the probability values must sum to one, the distribution described by the HMM is constrained. Recently, deep learning has emerged as a family of learning models that aim to model high-level abstractions in data, and deep learning-empowered HAR is enjoying considerable attentions due to its ability to learn deep structures of patterns [105]. Deep learning techniques have been shown to outperform well-established methods that rely on hand-crafted feature extraction and shallow feature learning architectures, owing to its ability to uncover features that are tied to the dynamics of human motion production, from simple motion encoding in lower layers to more complex motion dynamics in upper layers, thus scaling up activity recognition to more complex activities [106].

Among various deep learning models, *Deep Belief Networks* (DBN) use *Restricted Boltzmann Machines* (RBMs) for learning. By comparison, Recurrent Neural Networks (RNN) are capable of encoding/learning sequential information, thus offering more discriminative power over DBNs. Given a series of input signals $\{x_1, \ldots, x_n\}$, their temporal information can be extracted by LSTM [107] as follows:

$$
\begin{aligned}
i_t &= \sigma(W_1 x_t + W_2 h_{t-1}), \\
\tilde{c}_t &= \tanh(W_3 x_t + W_4 h_{t-1}), \\
f_t &= \sigma(W_5 x_t + W_6 h_{t-1}), \\
o_t &= \sigma(W_7 x_t + W_8 h_{t-1}), \\
c_t &= c_{t-1} \odot f_t + i_t \odot \tilde{c}_t, \\
h_t &= c_t \odot o_t,
\end{aligned}
\tag{4.1}
$$

where the subscript t denotes the time window, operator $\odot$ refers to component-wise multiplication, σ is the logistic sigmoid function, and h and c are the hidden state and the cell state respectively.

4.1 Distance-Preserving Multiplicative Perturbation

Euclidean distance-preserving data perturbation [72, 108] has been receiving attention as it delivers a promising trade-off between privacy and accuracy. Assume an organization owns a private, real-valued database $\mathbf{X}$ (row: features, column: data records) and wishes to make it publicly available for data analysis while keeping individual records private. To accomplish this, $\mathbf{Y} = \mathbf{TX}$ can be released to the public, where $\mathbf{T}$ preserves Euclidean distances between columns and is only known to the data owner. In this way, many common learning algorithms, with only minor modification, can be applied to $\mathbf{Y}$ and produce similar patterns that would be extracted from $\mathbf{X}$. Random projection preserves Euclidean distance and is especially suitable for high-dimensional data analysis. The reduced dimensions are not a subset of the original dimensions but rather a transformation, which is relevant for privacy preservation.

For a more rigorous analysis, let $\mathbf{T} = \left[t_{ij}\right]$, where $i = 1, \ldots, w$, $j = 1, \ldots, n$, and $w < n$. Given some distribution D, if $t_{ij} \sim \mathsf{D}$ is i.i.d., we write $\mathbf{T} \sim \mathsf{D}_{w \times n}$. Below, we consider two types of distributions (and hence two types of $\mathbf{T}$) in terms of their ability to preserve Euclidean distances and inner products in Proposition 4.1. Random projection has been proved to preserve Euclidean distances and inner products between data points in expectation [109, Appendix 5.6.1]. We further prove that uniform transformation using uniform distribution on the interval $[0, 1]$ does not preserve Euclidean distances and inner products between data points.

Proposition 4.1. *Suppose* $\mathbf{T}$ *is a* $w \times n$ *multiplicative perturbation matrix, where* $w < n$*. Then,*

Case 1 *If* $\mathbf{T} \overset{iid}{\sim} \mathsf{N}_{w \times n}(0, \sigma_t^2)$*, where* $\mathsf{N}_{w \times n}(0, \sigma_t^2)$ *is the zero-mean Gaussian distribution with variance* $\sigma_t^2 = \frac{1}{w}$*, then both the inner products and Euclidean distances are preserved in expectation, i.e., the error* $\boldsymbol{u}^\top \boldsymbol{v} - \boldsymbol{x}^\top \boldsymbol{y}$ *of the inner product produced by random projection is zero on average, and the variance is inversely proportional to the reduced dimensionality* w*. In particular, the variance is at most the inverse of the dimensionality of the reduced space multiplied by 2 if the original data vectors are normalised to unity [109, Appendix 5.6.1].*

Case 2 *If* $\mathbf{T} \overset{iid}{\sim} \mathsf{U}_{w \times n}(0, 1)$*, where* $\mathsf{U}(0, 1)$ *is the uniform distribution on the interval* $[0, 1]$*, then both the Euclidean distances and inner products between data points are not preserved in*

expectation, i.e., the error $\boldsymbol{u}^\top \boldsymbol{v} - \boldsymbol{x}^\top \boldsymbol{y}$ of the inner product produced by uniform transformation is not zero on average.

Proof. Let t_{ij} and ϵ_{ij} be the (i,j)-th entry of projection matrix $\mathbf{T}$ and $\mathbf{T}^\top\mathbf{T}$ respectively. We consider the two cases in turn:

Case 1 $t_{ij} \sim \mathsf{N}(0, \sigma_t^2) \implies \mathsf{E}[t_{ij}^2] = \sigma_t^2 = \frac{1}{w}$. It follows that $\mathsf{E}[\epsilon_{ii}] = w\sigma_t^2 = 1$, $\mathsf{Var}[\epsilon_{ii}] = 2w\sigma_t^4 = \frac{2}{w}, \forall i$; and $\mathsf{E}[\epsilon_{ij}] = 0$, $\mathsf{Var}[\epsilon_{ij}] = w\sigma_t^4 = \frac{1}{w}, \forall i \neq j$. Therefore, $\mathsf{E}[\mathbf{T}^\top\mathbf{T}] = \mathbf{I}$.

Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. Suppose $\boldsymbol{x}$ and $\boldsymbol{y}$ are projected to $\boldsymbol{u} = \mathbf{T}\boldsymbol{x}$ and $\boldsymbol{v} = \mathbf{T}\boldsymbol{y}$ respectively. Then,

$$\mathsf{E}[\boldsymbol{u}^\top \boldsymbol{v} - \boldsymbol{x}^\top \boldsymbol{y}] = \mathsf{E}[\boldsymbol{x}^\top \mathbf{T}^\top \mathbf{T}\boldsymbol{y} - \boldsymbol{x}^\top \boldsymbol{y}] = \boldsymbol{x}^\top\, \mathsf{E}[\mathbf{T}^\top\mathbf{T}]\boldsymbol{y} - \boldsymbol{x}^\top \boldsymbol{y} = 0,$$

$$\mathsf{Var}[\boldsymbol{u}^\top \boldsymbol{v} - \boldsymbol{x}^\top \boldsymbol{y}] = \frac{1}{w}\left\{\sum_i x_i^2 \sum_i y_i^2 + \left(\sum_i x_i y_i\right)^2\right\}.$$

In particular, if both $\boldsymbol{x}$ and $\boldsymbol{y}$ are normalised, then $(\sum_i x_i^2)(\sum_i y_i^2) = 1$, $(\sum_i x_i y_i)^2 \leq 1$, and the upper bound of the variance becomes:

$$\mathsf{Var}\left[\boldsymbol{u}^\top \boldsymbol{v} - \boldsymbol{x}^\top \boldsymbol{y}\right] \leq \frac{2}{w},$$

which is inversely proportional to the reduced dimension w. Applying the results above to vectors $\boldsymbol{u} - \boldsymbol{v}$ and $\boldsymbol{x} - \boldsymbol{y}$, we have

$$\mathsf{E}\left[(\boldsymbol{u} - \boldsymbol{v})^\top (\boldsymbol{u} - \boldsymbol{v}) - (\boldsymbol{x} - \boldsymbol{y})^\top (\boldsymbol{x} - \boldsymbol{y})\right] = 0 \implies \mathsf{E}\left[\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 - \|\boldsymbol{x} - \boldsymbol{y}\|_2^2\right] = 0.$$

If both $\boldsymbol{x}$ and $\boldsymbol{y}$ are normalised, then $\mathsf{Var}\left[\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 - \|\boldsymbol{x} - \boldsymbol{y}\|_2^2\right] \leq \frac{32}{w}$.

Case 2 $t_{ij} \sim \mathsf{U}(0,1) \implies \mathsf{E}[t_{ij}] = \frac{1}{2}$, and $\mathsf{E}[t_{ij}^2] = \frac{1}{3}$. As $\epsilon_{ii} = \sum_{k=1}^w t_{ki}^2$ and $\epsilon_{ij} = \sum_{k=1}^w t_{ki} t_{kj}, \forall i \neq j$, thus $\mathsf{E}[\epsilon_{ii}] = \mathsf{E}\left[\sum_{k=1}^w t_{ki}^2\right] = w\,\mathsf{E}[t_{ki}^2] = \frac{w}{3}$, and $\mathsf{E}[\epsilon_{ij}] = \mathsf{E}\left[\sum_{k=1}^w t_{ki} t_{kj}\right] = \sum_{k=1}^w \mathsf{E}[t_{ki}]\,\mathsf{E}[t_{kj}] = \frac{w}{4}$. Therefore,

$$\mathsf{E}[\mathbf{T}^\top\mathbf{T}] = [\tau_{ij}], \quad \text{where } \tau_{ij} = \begin{cases} \frac{w}{4}, & \text{if } i \neq j; \\ \frac{w}{3}, & \text{if } i = j. \end{cases}$$

It then follows that

$$\mathsf{E}[\boldsymbol{u}^\top \boldsymbol{v} - \boldsymbol{x}^\top \boldsymbol{y}] = \mathsf{E}[\boldsymbol{x}^\top \mathbf{T}^\top \mathbf{T}\boldsymbol{y} - \boldsymbol{x}^\top \boldsymbol{y}] = \boldsymbol{x}^\top \mathsf{E}[\mathbf{T}^\top \mathbf{T}]\boldsymbol{y} - \boldsymbol{x}^\top \boldsymbol{y} = \boldsymbol{x}^\top \boldsymbol{y}(E[\mathbf{T}^\top \mathbf{T}] - 1) \neq 0.$$

Hence, both the Euclidean distances and inner products between data points are not preserved. □

Proposition 4.1 shows that a normally distributed RP matrix preserves both Euclidean distances and inner products in expectation, hence it is beneficial for data analytics such as clustering and classification. In terms of resistance to different attacks, at the first stage, we rely on a nonlinear function to resist MAP estimation attack, and at the second stage, we use random projection to map the transformed data to a low-dimensional subspace to resist ICA attacks .

4.2 Privacy Analysis

In this section, we first describe the developed RG+RP scheme, and then discuss its robustness to MAP estimation attack and ICA attack.

4.2.1 RG+RP Scheme

In the setup phase, each participant c_i is given a $w \times n$ ($w < n$) random projection matrix, denoted $\mathbf{T}$, which is generated i.i.d. from Gaussian distribution $N(0, 1/w)$. Suppose participant c_i is contributing data $\mathbf{X}_i \in \mathbb{R}^{n \times m_i}$ to the cloud service S for deep learning. The participant transforms $\mathbf{X}_i$ to $\mathbf{Z}_i \in \mathbb{R}^{w \times m_i}$ in two stages:

Stage 1: The participant transforms $\mathbf{X}_i$ to $\mathbf{Y}_i$, by applying the nonlinear perturbation function N element-wise:

$$\mathbf{Y}_i = N(\mathbf{X}_i). \tag{4.2}$$

N is chosen to be the repeated Gompertz function, as illustrated in Figure 3.2.

Stage 2: Using the random projection matrix $\mathbf{T}$ initialized in the setup phase, the participant transforms $\mathbf{Y}_i$ to $\mathbf{Z}_i$ as follows:

$$\mathbf{Z}_i = \mathbf{T}\mathbf{Y}_i. \tag{4.3}$$

The participant then sends $\mathbf{Z}_i$ to the cloud service $\mathcal{S}$. Once $\mathcal{S}$ receives all the perturbed datasets $\mathbf{Z}_i$, $i = 1, \ldots, q$, it concatenates them as: $\mathbf{Z}_{\text{all}} = [\mathbf{Z}_1 | \cdots | \mathbf{Z}_q]$, then builds a deep learning model on $\mathbf{Z}_{\text{all}}$. The pseudocode for the collaborative deep learning procedure is presented in Algorithm 4.1. The recognition results can be used by end-users for further analysis or to provide valuable feedback to the participants.

Algorithm 4.1 The collaborative deep learning procedure

Step: Participant
$\mathbf{T} \leftarrow$ a $w \times n$ random projection matrix generated from Gaussian distribution $N(0, 1/w)$
$\mathbf{Z}_i \leftarrow \mathbf{T}(N(\mathbf{X}_i))$
Send $\mathbf{Z}_i$ to the cloud service

Step: Cloud service
Receives $\mathbf{Z}_i$, $i = 1, \ldots, q$
$\mathbf{Z}_{all} \leftarrow \left[\mathbf{Z}_1 | \mathbf{Z}_2 | \cdots | \mathbf{Z}_q\right]$
Build deep learning model on $\mathbf{Z}_{all}$
Send recognition results to the end-users

Step: End-user
Receive recognition results from the cloud service

4.2.2 Privacy Evaluation

This section presents the simulation and evaluation results of RG+RP in terms of its recovery resistance. Since RG+RP is inherently resistant to ICA attack, the empirical experiments focus on the recovery rates of MAP estimation attack against RG+RP, and compare with prior work, *i.e.*, [80], [85] and [83]. Experiments are conducted on both synthetic and real datasets shown in Table 4.1.

In order to evaluate the recovery resistance of RG+RP against MAP estimation attack, experimental results are provided in terms of the ϵ-recovery rate defined in Eq. (3.5). We estimate the ϵ-recovery rate as the fraction of test data that can be recovered to within a relative error of ϵ.

MAP estimation attack is simulated in the same way as in Sect. 3.3.1. The schemes shown in Table 4.2 are evaluated in the *worst-case* scenario where the attacker knows exactly the victim's perturbation matrix. The results for different datasets are as follows.

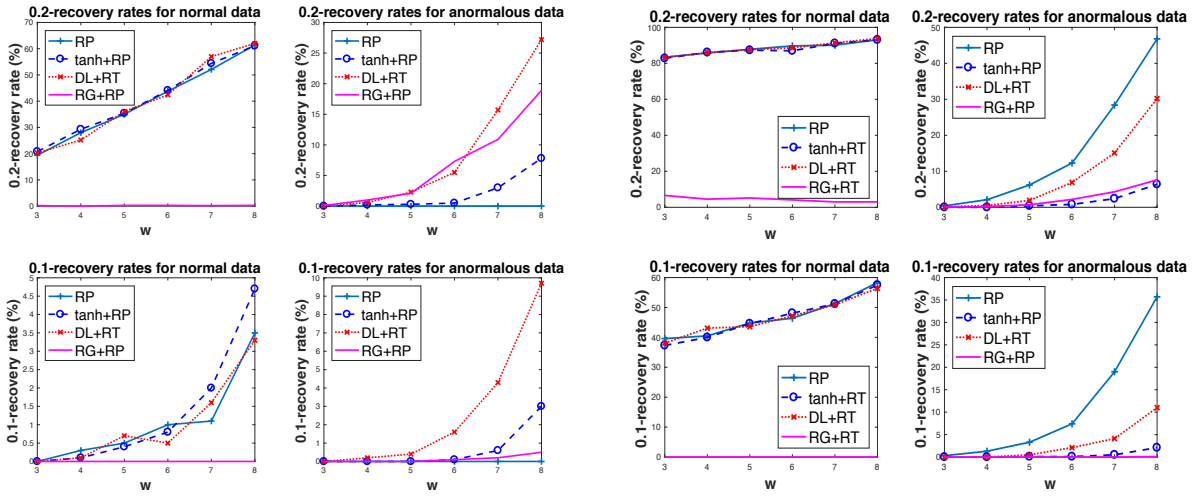
Purely Gaussian datasets: For Gaussian datasets, Figure 4.1 (a) shows that RG+RP provides significantly higher recovery resistance for both normal and anomalous data

Table 4.1: Datasets for evaluating recovery resistance.

Datasets	#Records(m)	Upspace dimension (n)	Downspace dimension (w)
Purely Gaussian	5000	15	8
Purely Laplace	5000	15	8
Abalone	4177	8	4
Forest	581012	54	27
Adult	48842	123	62
Gas	13910	128	64
OAR	77597	110	55
DSA	7500	315	158
HAR	7352	561	281
Smiley	20000	20	10
GME	101000	100	50

Table 4.2: Evaluated schemes.

Scheme	Nonlinear perturbation function (stage 1)	Linear projection matrix (stage 2)
RP [80]	none	$\mathbf{T} \sim N_{w\times n}(0,4)$
tanh+RP	tanh [85]	$\mathbf{T} \sim N_{w\times n}(0,1)$
DL+RT [83]	double logistic	$\mathbf{T} \sim U_{w\times n}(0,1)$
RG+RP [38]	repeated Gompertz	$\mathbf{T} \sim N_{w\times n}(0,1/w)$



(a) Gaussian-distributed data.

(b) Laplace-distributed data.

Figure 4.1: Recovery rates of MAP estimation attack against evaluated schemes, on $w \times 1000$ data projected from 15×1000 normalised Gaussian-distributed data (zero mean, identity covariance matrix) and Laplace-distributed data (zero mean, unity scale).

compared to the other schemes, except for the 0.2-recovery rate for the anomalous data case, which is slightly less effective than RP. An intuitive explanation for why RP excels

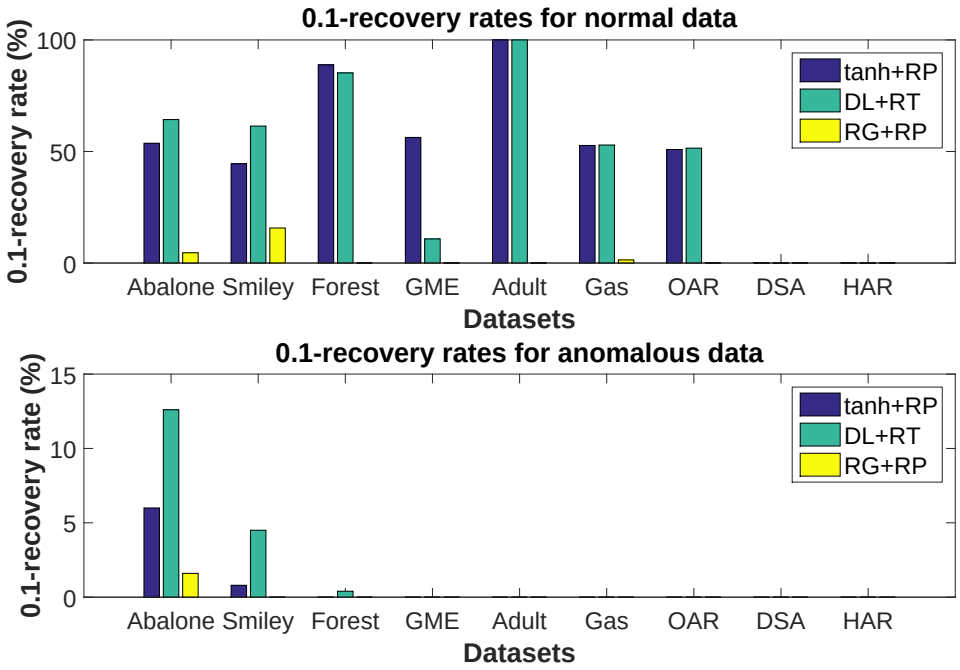


Figure 4.2: 0.1-recovery rates of MAP estimation attack against the evaluated schemes, on various datasets. The rank of the perturbation matrix, $w = \lfloor (n+1)/2 \rfloor$, where n is the number of features. Note zero recovery rates in many cases.

at protecting anomalous data is that RP has the effect of "normalising" anomalous data in the range space of $\mathbf{T}$.

Purely Laplace datasets: Figure 4.1 (b) demonstrates that RG+RP significantly outperforms other methods for Laplace datasets, and it is especially evident for normal data, except for the 0.2-recovery rate for the anomalous data case, which is slightly less effective than tanh+RP. Furthermore, the 0.1-recovery rate against RG+RP is below 10%, which is significantly lower than other schemes.

Assorted real and synthetic datasets: Consistent with the results for purely Gaussian and purely Laplace datasets, as shown in Figure 4.2, RG+RP also outperforms tanh+RP and DL+RT in terms of recovery resistance for both normal data and anomalous data. It delivers low recovery rates in many cases, especially for DSA and HAR datasets, RG+RP achieves (0.1, 0)-recovery resistance.

4.3 Utility Analysis

For utility analysis, we apply different models to both original and perturbed HAR and MH datasets to study the effect of the privacy-preserving scheme. Moreover, we experimentally compare the proposed LSTM-CNN model with three baseline models in terms of recognition accuracy.

4.3.1 LSTM-CNN Model

In deep learning models, CNNs and LSTMs are complementary in their modeling capabilities, as CNNs aim at reducing frequency variations, while LSTMs are especially good for temporal modeling. It is explored in [110] that RNNs outperform CNNs significantly on activities that are short in duration but have a natural ordering, however, CNNs are more suitable for prolonged and repetitive activities like walking or running. Therefore, it would be beneficial to exploit the synergy of CNNs and LSTMs by combining them into one unified architecture and train them jointly. In our architecture, we leverage the local and dense property from convolution operation and learn the temporal structure by storing information in LSTM units, a CNN layer is put above LSTM layer, as illustrated in Figure 4.3. First, the input features are fed into LSTM layers to reduce temporal variation. At each time step, the LSTM is capable of combining the previous information with the current input. Afterwards, we concatenate these outputs to a matrix $\mathbf{A} \in \mathbb{R}^{k \times s}$, where s denotes the number of time steps and k denotes the number of features.

Motivated by Blunsom *et al.* [111], we introduce convolutional layers applying one-dimensional filters across each column of time signals on the matrix $\mathbf{A} \in \mathbb{R}^{k \times s}$. We believe convolving the same filter can extract the dominant information over s consecutive time signals. In order to capture different positional information, multiple filters with varying size are utilized. Each convolution operation involves a filter $\mathbf{w} \in \mathbb{R}^{k \times m}$, which is applied to a window of m columns to produce a new feature. For example, a feature c_i can be generated from Eq. (4.4):

$$c_i = \mathrm{ReLU}(\langle \mathbf{w}, \mathbf{A}_{i:i+m-1} \rangle + b), \tag{4.4}$$

where

- ReLU is the Rectified Linear Unit activation function;
- $\mathbf{A}_{i:i+m-1}$ refers to $\{\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_{i+m-1}\}$, for $\boldsymbol{x}_i \in \mathbb{R}^k$, $i = \{1, \ldots, s-m+1\}$;
- $\langle \mathbf{A}, \mathbf{B} \rangle$ denotes the Frobenius inner product of $\mathbf{A}$ and $\mathbf{B}$;
- $b \in \mathbb{R}$ is a bias term.

Accordingly, for each filter, we can obtain a sequence of features: $c_1, c_2, \ldots, c_{s-m+1}$.

Like other CNN structures, the convolutional layer is followed by a maxpooling layer, which aims at capturing the most crucial features in a specific feature mapping, as indicated in Eq. (4.5):

$$c_{max} = \max(c_1, c_2, \ldots, c_{s-m+1}). \tag{4.5}$$

Once we obtain all predominant features from CNN, multi-layer perceptron (MLP) can be used to conduct activity classification.

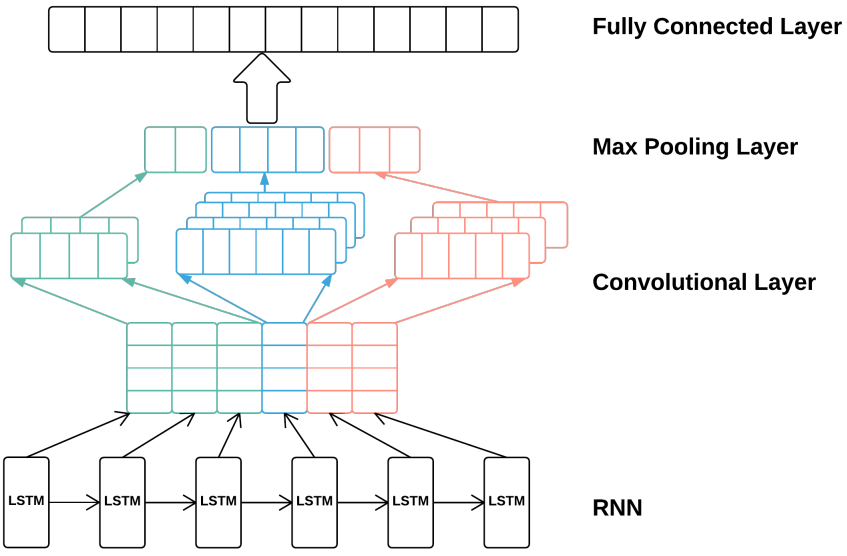


Figure 4.3: LSTM-CNN architecture. The bottom layer is a RNN nets with 6 time steps. At convolutional layer, we have 4 unigram filters, and 3 bigram filters and 2 trigram filters. Then maxpooling layer concatenates all maximum features of feature maps. Finally these features are fed into a MLP to perform activity classification.

4.3.2 Wearable Datasets

The initial accuracy evaluation of the proposed scheme is based on two real-world wearable datasets from the UCI Machine Learning Repository. The first wearable dataset is Human Activity Recognition database [103], which consists of recordings of 30 subjects performing activities of daily living (ADL). Each person performed six activities by wearing a waist-mounted smartphone (Samsung Galaxy S II) with embedded inertial sensors. From the embedded accelerometer and gyroscope, 3-axial linear acceleration and 3-axial angular velocity were captured at a constant rate of 50Hz. The labels were recorded by video. The sensor signals were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). From each window, a vector of 561 features was obtained by calculating variables from the time and frequency domain.

Another wearable dataset is the Mobile Health (MH) dataset [112], which comprises body motion and vital signs recordings for ten volunteers while performing 12 common activities. Sensors placed on the subject's chest, right wrist and left ankle were used to measure the motion experienced by diverse body parts, namely, acceleration, rate of turn and magnetic field orientation. The sensor positioned on the chest also provides 2-lead ECG measurements, which can be potentially used for basic heart monitoring, checking for various arrhythmias or looking at the effects of exercise on the ECG. All sensing modalities were recorded using a video camera at a sampling rate of 50 Hz, and sampled in fixed-width sliding windows of 128 readings/window.

Both HAR dataset and MH dataset have been randomly partitioned into three sets, 70% training data, 15% validation data and 15% test data. We choose the epoch that delivers the best validation set performance for testing purpose.

4.3.3 Model Evaluation

For accuracy evaluation, we first establish baseline models, namely CNN, LSTM and DBN. The recognition accuracy for the raw data and RG+RP perturbed data are referred to as *Raw* and *Perturbed* respectively. The corresponding accuracy results for HAR and MH datasets are shown in Table 4.3.

Table 4.3: Comparing the proposed LSTM-CNN model with three baseline models in terms of recognition accuracy, using the HAR and MH datasets. The best results are highlighted in bold.

Model	HAR		MH	
	Raw	Perturbed	Raw	Perturbed
CNN	0.9225	0.9033	0.8399	0.8068
LSTM	0.9134	0.8048	0.8233	0.7996
DBN	0.9536	0.8455	0.8815	0.8237
LSTM-CNN	**0.9844**	**0.9375**	**0.9556**	**0.9208**

Table 4.4: Accuracy of the proposed LSTM-CNN for HAR dataset under different hyper-parameters. For CNN filters, (a,b) denotes b a-gram filters.

Model	HAR			
	Raw		Perturbed	
Input Features	9	9	9	9
Hidden Units for LSTM	18/28	18/28	18/28	18/28
Layers for LSTM	1	2	1	2
CPU time per Mini-Batch	38s/40s	79s/85s	20s/25s	48s/53s
Filters for CNN	(30, 5), (40, 10), (50, 15), (60, 20)			
Accuracy	0.970/0.984	0.980/0.988	0.931/0.937	0.933/0.938

For HAR dataset, we find that the LSTM-CNN architecture provides a 7.77% relative improvement over the LSTM, 6.71% relative improvement over the CNN, and 3.23% relative improvement over the DBN. We also notice that the proposed privacy-preserving RG+RP mechanism achieves 93.75% accuracy, which lowers the accuracy by only 4.76% compared to the result from the raw data, and is even higher than the accuracy delivered by the LSTM and CNN models trained on the raw data.

For MH dataset, the LSTM-CNN architecture exhibits a similar accuracy trend, delivering an accuracy higher than 92% for both the raw data and perturbed data. Considering the significant privacy benefits provided by our architecture, the accuracy is competitive. A plausible explanation for the comparable accuracy is that RP has negligible impact on accuracy degradation, and RG is designed to be nearly consistent in trend with the original data, thus causing small performance degradation. In addition, unlike simple linear projections, which can't capture intrinsic nonlinearities due to limited representational power, we hypothesize that non-linear projections make data closer to linearly separable and thus easier to classify. Table 4.4 and Table 4.5 show the results of varying

Table 4.5: Accuracy of the proposed LSTM-CNN for MH dataset under different hyper-parameters. For CNN filters, (a,b) denotes b a-gram filters.

Model	MH			
	Raw		Perturbed	
Input Features	21	21	21	21
Hidden Units for LSTM	28/64	28/64	28/64	28/64
Layers for LSTM	1	2	1	2
CPU time per Mini-Batch	39s/43s	84s/90s	28s/31s	52s/59s
Filters for CNN	(30, 5), (40, 10), (50, 15), (60, 20)			
Accuracy	0.950/0.955	0.958/0.962	0.915/0.920	0.918/0.921

LSTM depth. Deeper models outperform shallower ones by a small margin at the cost of doubling the training time. Similarly, larger hidden units do not significantly improve accuracy.

4.4 Summary

In this chapter, we consider the scenario where a centralized server performs deep learning on the collected data contributed by a large number of participants. To remove the deterrents for users to share their data and benefit from the community knowledge discovery afforded by PS/MCS, service providers need privacy-preserving algorithms that deliver a reasonable trade-off between data privacy and utility. Hence the problem we are addressing is *how a data owner can release its data with the guarantees that the original sensitive information cannot be reidentified while the analytic properties of the data are preserved*. For a deep learning approach, we apply LSTM-CNN to human activity recognition from wearable sensor data. Compared with traditional DBN models and standalone LSTM, CNN models, our analysis explicitly demonstrates the competitive performance of LSTM-CNN in dealing with time-series data. For privacy preservation, we propose a revised two-stage scheme called RG+RP, where in the first stage, each participant perturbs its data by passing the data through a nonlinear function called *repeated Gompertz* (RG); and in the second stage, each participant's data is projected to a lower dimension using Gaussian *random projection* (RP) matrix to ensure the accuracy of time-series classification, especially for large data size. The nonlinear function is designed to condition the probability

density function (pdf) of the perturbed data to thwart *maximum a posteriori* (MAP) estimation attack, whereas RP compresses the data in a manner that stochastically preserves the Euclidean distances between pairs of data points as per the Johnson-Lindenstrauss Lemma to maintain utility and be resistant to ICA attack. For evaluation, we use ϵ-*recovery rate* to assess how much privacy is preserved and how likely users' private data can be compromised, and use recognition accuracy to measure the utility of our privacy-preserving scheme.

Genius is one percent inspiration, ninety-nine percent of blood and sweat.

Thomas Edison

5

Decentralized Privacy-Preserving Machine Learning in Biomedical Domain

Large-scale models trained on datasets distributed over multiple parties offer clear advantages over analysis restricted to a single organisation's homogeneous data source of limited size. A common case is horizontally-partitioned datasets, wherein each party curates a separate groups of individual records represented by a feature set common to all parties. For example, the biomedical domain is experiencing increasing digitisation of many types of data—demographic, clinical, and genomic—often collected and stored by independent hospitals or medical research institutions [13].

Privacy implications of sharing such sensitive information as medical data, places limitations on more widespread use [113]. Even trained models or their predictions can reveal training data through black-box attacks [14, 58]. Therefore, neither training data nor derived learned models and predictions can be directly shared with untrusted third parties, motivating the need for distributed privacy-preserving algorithms [114]. Ex-

isting state-of-the-art privacy-preserving collaborative frameworks are either based on computationally intensive techniques *e.g.*, [50] or rely on a central server to distribute key shares, mediate the modelling process, or aggregate locally trained models [8–10]. Unfortunately server-based frameworks present security and robustness vulnerabilities: if the central server gets compromised the entire network is under risk of compromise. To address these limitations it is desirable to remove the centralised server and distribute computation among the constituent parties.

In this chapter, we consider distributed differential privacy while learning a centroid classifier, which corresponds to the support vector machine (SVM) in a limiting regularisation regime, and which has demonstrated comparable performance to the SVM on microarray data [115]. Differentially-private release of SVM classifiers has been previously considered under the single-party setting [116]. We build a *decentralized privacy-preserving centroid classifier* (DPPCC) for horizontally partitioned data. DPPCC distinguishes itself from existing techniques by making no use of trusted and centralised third party aggregators. We achieve decentralisation and privacy through the use of exponential ElGamal and differential privacy, and obtain performance competitive with the equivalent non-private classifier. Using this model we investigate three practical sharing scenarios where all parties collaborate, without disclosure of their data, to:

1. learn a shared model;
2. produce a prediction for a given sample; and
3. produce a prediction without revealing the sample.

5.1 Distributed Learning Approaches

In general, distributed learning approaches fall into three categories of aggregation: trusted aggregator, untrusted aggregator, or no aggregator.

Trusted aggregator. Traditional techniques consider a trusted third party who is entitled to see all participants' data in the clear. The pure differential privacy framework [86] assumes a centralised data custodian.

Untrusted aggregator. By contrast, an untrusted aggregator is not trusted by any

parties, hence parties must sufficiently blind shared statistics of input data, such as via randomisation in guaranteeing *Local Differential Privacy* (LDP). In the case of additive noise, aggregation sums up individual noise shares such that the final aggregation only preserves utility under very large population sizes. To preserve privacy and maintain utility (especially under the more challenging case of modest data sizes), differential privacy can be made distributed by combination with cryptographic protocols, as evidenced by the following schemes.

Rastogi *et al.*[9] leverage an untrusted aggregator to aggregate sums over multiple data sources using the Paillier semi-homomorphic encryption scheme. Their proposed *Distributed Laplace Perturbation Algorithm* generates Laplace noise using four Gaussian variables, while our solution is more efficient, by exploiting stability of the simple discrete Gaussian distribution. Furthermore, we remark that their scheme is not resistant to collusion attacks as malicious users can collude with the aggregator to infer information about honest users—implying that the aggregator may not be fully untrusted. In the pathological case of an untrusted aggregator colluding with the remaining $N-1$ parties, they might try to retrieve a victim party's data x_i by decrypting and subtracting the other $N-1$ parties' values from the result. A detailed description of flaws in their scheme, witnessed by specific attacks, is provided in Appendix A.3.

Ács *et al.*[8] propose a differential-privately summation over smart meter data, over multiple time slots, where the aggregator is untrusted. Smart meters are grouped into clusters, where a cluster covers hundreds or thousands of smart meters corresponding to a quarter of a city and Laplace-perturbed readings are sent to an electricity distributor. This requires all meters in a cluster to share pairwise keys.

Shi *et al.*[10] propose *distributed differential privacy* (DDP) to guarantee (ϵ, δ)-differential privacy of the aggregate of time-series data in every time period, where noise is sourced from multiple participants, and no parties (or the aggregator) should view anything but the final differentially-private result. They use symmetric geometric distribution as a discrete approximation to the Laplace distribution, where each participant adds noise probabilistically. However, their encryption scheme relies on a trusted dealer allocating $q+1$ secrets that sum to 0, to the data aggregator and the q participants; also unfortunately their construction is not robust against node and communication failures.

No aggregator. In the absence of an aggregator, Dwork *et al.* [50] first propose a distributed protocol called "Our Data, Ourselves" to generate shares of random noise, secure against malicious participants. The shares of random binomial noise are generated by coin flipping, which is secure against malicious participants. However, this requires communication among users and the expensive secret sharing technique results in $O(n)$ multiplications and additions in shares, where n denotes the number of participants, thus not preferable for large number of users.

In this chapter, we propose a purely decentralized aggregation model, which provides fault tolerance and eliminates the aggregator by distributing noise generation and aggregation among existing parties.

5.2 Preliminaries

5.2.1 Centroid Classifier

Let $\mathbf{X}$ be an $m \times d$ data matrix of m training instances in $\mathbb{R}^d$ and $\boldsymbol{Y} \in \mathbb{R}^m$ be the vector of corresponding labels in $\{-1, 1\}$. The labelled training set right-augmenting $\mathbf{X}$ by column vector $\boldsymbol{Y}$ is denoted by $m \times (d+1)$ matrix $\mathbf{D} \in (\mathbb{R}^d \times \{-1, 1\})^m$.

It is well supported from the perspective of statistical learning theory that regularisation can be used to generate robust empirical estimators and classifiers from noisy data [117–119]. An limiting case of regularisation of SVMs corresponds to the centroid classifier, where one represents the positive and negative classes by their centroids, and performs classification based on the distance of an instance to these two centroids. The centroid classifier is defined on an instance x as:

$$g_B(x) = \frac{1+B}{2} \sum_{i:y_i=+1} \frac{k(x, x_i)}{m_{+1}} - \frac{1-B}{2} \sum_{i:y_i=-1} \frac{k(x, x_i)}{m_{-1}} + b$$

where $k(\cdot, \cdot)$ is a kernel, b is a learned bias term, and $-1 \leq B \leq 1$ is a hyperparameter representing misclassification cost ratio between the two classes. The feature space is the projection onto the vector connecting (weighted) arithmetic means of samples from both class labels, degenerating to the mean of a single class for $B = \pm 1$, respectively [115]. For a suitable kernel k, it implements the difference between Parzen window estimates

of probability densities for both labels. Under the linear kernel, it simplifies to:

$$g_B(x) = \boldsymbol{w_B} \cdot \boldsymbol{x} + b := \left(\frac{1+B}{2} \bar{\boldsymbol{x}}_{+1} - \frac{1-B}{2} \bar{\boldsymbol{x}}_{-1} \right) + b$$

where $\boldsymbol{w_B} \in \mathbb{R}^n$ represents the centroid weight vector and $\bar{\boldsymbol{x}}_y$ stands for the centroid of the samples with label $y = \pm 1$. If $B = 0$ and $b = \boldsymbol{w_B} \cdot (\bar{\boldsymbol{x}}_{+1} + \bar{\boldsymbol{x}}_{-1})/2$, the decision hyperplane will lie exactly halfway between $\bar{\boldsymbol{x}}_{+1}$ and $\bar{\boldsymbol{x}}_{-1}$. If we weigh each class's contribution to costs equally under unbalanced classes, via $B = (m_{+1} - m_{-1})/m$, where m_{+1} and m_{-1} are the positive and negative class sizes, $m := m_{+1} + m_{-1}$, then this corresponds to van Rooyen *et al.'s* unhinged loss function $-yf(x)$ with solution $\boldsymbol{w} = \mathbf{X}^\intercal \mathbf{Y}/m$ [120].

5.2.2 Distributed Differential Privacy

Dwork *et al.*[86] first proposed the privacy criterion of *differential privacy* for the single database scenario, where for every query, a privacy-preserving mechanism randomises the response to the query. For the scenario where data are sourced from multiple parties who do not trust the aggregator or without any aggregator, a distributed version of differential privacy—*distributed differential drivacy* (DDP)—is needed [8–10,50]. DDP reflects the fact that the noise in the target statistic is sourced from multiple parties for guaranteeing differential privacy of the final result, while also guaranteeing that no intermediate (non-private) computations are observed by the collaborating parties.

Shi *et al.* [10] first defined DDP for the scenario where a single aggregator evaluates an aggregation function; we reformulate DDP for decentralized aggregation.

Each participant i possesses a share of data $\boldsymbol{x}_i \in \mathbb{R}^d$. If honest, the participant obtains random bits $\boldsymbol{r}_i \in \{0,1\}^q$, which together with its data, it maps to randomised statistic $\hat{\boldsymbol{x}}_i$ via function f_i. All parties collaborate with intended goal to evaluate some $\chi : (\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_n) \mapsto s$ such that the composition $\chi \circ (f_1, \ldots, f_n) = \mathcal{M}$ some desired randomised mechanism mapping database $D \in \mathcal{D}^n$ to response space $\mathcal{R}$.

Given a subset K of parties, we let $\boldsymbol{r}_K := \{\boldsymbol{r_i} : i \in K\}$ and denote $\bar{K}$ be the complement of K, *i.e.*, $\bar{K} = \{1, 2, ..., n\} \setminus K$.

Two databases D, D' are said to be neighbours or neighbouring databases if they differ in exactly one record. This corresponds to one party changing a record.

Definition 5.1 (Distributed DP). *Let $\epsilon > 0$, $0 \leq \delta < 1$ and $0 < \gamma < 1$, We say that the mechanism $\mathcal{M}$ as described above, with randomness over the joint distribution of $\boldsymbol{r} := (\boldsymbol{r_1}, \cdots, \boldsymbol{r_n})$ preserves $(\epsilon, \delta, \gamma)$-distributed differential privacy (DDP), if the following conditions hold. (1) For any neighbouring databases $D, D' \in \mathcal{D}^n$ that differ in one record, for any measurable $S \subseteq \mathcal{R}$ response space of $\mathcal{M}$, and for any subset $\bar{K}$ of at least γn honest participants,*

$$\Pr\left(\mathcal{M}(D) \in S | \boldsymbol{r}_K\right) \quad \leq \quad \exp(\epsilon) \cdot \Pr\left(\mathcal{M}(D') \in S | \boldsymbol{r}_K\right) + \delta.$$

That is if a sufficient number of parties the mechanism completes with a release $s \in \mathcal{R}$ that preserves approximate DP. (2) No party, irrespective of whether it is honest or not, can learn anything about any other party's data share except via the final release $s \in \mathcal{R}$—semantically secure collaboration χ.

When K is the set of compromised parties, the above definition requires that the remaining honest parties' randomness be sufficient to ensure differential privacy. Therefore, the probability is conditioned on the randomness $\boldsymbol{r}_K$ from compromised parties. In other words, the probability is taken over the randomness $\boldsymbol{r}_{\bar{K}}$ from honest parties. The definition of DDP requires smoothness to hold over any set $\bar{K}$ of uncompromised parties, for $|\bar{K}| \geq \gamma n$. For differentially-private aggregation of local statistics, DDP permits each individual to randomise its local statistic to a lesser degree than would local differential privacy. Provided the final randomised release by $\mathcal{M}$ has accumulated sufficient randomness, each individual's privacy is guaranteed.

Approaches to DDP that implement an overall additive noise mechanism by summing the same mechanism run at each party (typically with less noise) necessitates mechanisms with stable distributions—to guarantee proper calibration of known end-to-end response distribution—and cryptography for hiding all but the final result from participating parties [8–10, 50].

5.2.3 Distributed Exponential ElGamal Cryptosystem

As the released randomised plain text space for each party is expected to be small, namely the weight vector that depends on the number of features or only a scalar test result, we employ exponential ElGamal [64]. Instead of encrypting m in the traditional ElGamal,

ElGamal is used in an additive manner here by encrypting g^m, here $g \in G_q$ is an arbitrary, publicly known element that generates the public key. The difficulty in decryption lies in the computation of discrete logarithms, as running the standard decryption only yields g^m, and recovering m requires solution of the discrete log. However, as long as m is small, this can be solved either algorithmically, or through a pre-computed lookup table. It has been shown that ElGamal is semantically secure, *i.e.*, it is computationally infeasible to distinguish between the encryptions of any pair of the given messages, provided that the decisional Diffie-Hellman problem is intractable [121]. This makes the scheme perfectly suited to DDP. The procedure for Exponential ElGamal can be illustrated as follows:

1. **Key generation:** Let p and q be large primes so that q divides $p-1$. Let g be a generator of G_q, which is a multiplicative subgroup of ${Z_p}^*$ of order q. The private key is $x \in Z_q$; the public key is $h = g^x$.

2. **Encryption:** A plaintext $m \in G_q$ is mapped onto an element g^m of G. g^m is encrypted by computing the following ciphertext tuple where $r \in Z_q$ is an arbitrary random number chosen by the encrypter:

$$(c_1, c_2) = (g^r, g^m h^r)$$

3. **Decryption:** g^m is decrypted by computing:

$$c_2 \cdot c_1^{-x} = g^m h^r \cdot g^{-xr} = g^m$$

In the distributed exponential ElGamal cryptosystem, key generation is distributed among n parties. Each party independently generates a standard key pair based on the public parameters, namely the group description (G, q, g, p) where: p is prime, G is a cyclic group of order q, and g is a generator of the group. All computations in the remainder of this chapter are modulo p unless otherwise noted. To achieve sharing, each party publishes its public key, then combines all the public keys to form a single joint public key. Anything encrypted with the joint public key can only be decrypted if all the participating parties each perform a partial decryption using the corresponding secret key. Those partial decryptions can be publicly shared and the final decryption can

be performed publicly—-assuming the final result could be public, otherwise it should be done privately. Our proposed distributed exponential ElGamal encryption consists of four components: distributed key generation, encryption, partial decryption and final decryption. The detailed procedure for parties Alice, Bob and Lily is described in Algorithm 5.1.

Algorithm 5.1 Distributed exponential ElGamal encryption

Distributed key generation

1: Alice, Bob and Lily generate an efficient description of a cyclic group G of order q, with generator g.
2: Alice, Bob and Lily select their private key shares $x_i \in {Z_q}^*$, which falls into $\{1, \ldots, q-1\}$.
3: Alice, Bob and Lily individually compute $h_i = g^{x_i}$, along with the description of (G, q, g, p) as their public keys, which are published to other parties. Each party keeps x_i as its private key, which must be kept secret.
4: Alice, Bob and Lily receive public key h_i from each other, the key pair of the cryptosystem corresponds to:
joint public key: $h = h_1 \cdot h_2 \cdot h_3 = g^{x_1+x_2+x_3}$
secret key: $x = \sum_{i=1}^{3} x_i$.

Encryption

1: Each party encrypts its message m_i (in case 1, $m_i = \hat{\boldsymbol{w}}_i$, and in case 2 and case 3, $m_i = \hat{y}_i$, here $\hat{}$means DP perturbed version, see Section 5.4.4) under the joint public key (G, q, g, p, h).
2: Each party chooses a random r from Z_q, then calculates $c_1 := g^r$.
3: Each party maps secret message m_i onto an element m_i' in G by computing ${m_i}' = g^{m_i}$.
4: Each party encrypts ${m_i}'$ with the joint public key as $c_2 := {m_i}' \cdot h^r$.
5: Each party sends ciphertext $(c_1, c_2) = (g^r, {m_i}' \cdot h^r)$ to other parties.
6: Once one party receives encryptions from other parties, it combines all the encryptions as $(c_1, c_2) = (g^r, m' \cdot h^r)$, where $m' = g^m$ (in case 1, $m = \hat{\boldsymbol{w}}_1 + \hat{\boldsymbol{w}}_2 + \hat{\boldsymbol{w}}_3$, and in case 2 and case 3, $m = \hat{y}_1 + \hat{y}_2 + \hat{y}_3$).

Partial decryption

1: Each party uses its secret key to compute the corresponding partial decryption of m' and forwards it back to other parties.

Final decryption

1: Denote each party's shared secret as $s_i := c_1^{x_i}$, after combing all parties' shared secrets, $s := c_1^{x_1+x_2+x_3}$. The requested party completes final decryption by computing $c_2 \cdot s^{-1} = m' \cdot h^r \cdot c_1^{-(x_1+x_2+x_3)} = m' \cdot g^{(x_1+x_2+x_3)r} \cdot g^{-r(x_1+x_2+x_3)} = m'$ where s^{-1} is the inverse of s in the group G.
2: m' is reverted back into the plaintext message m using look up table.

With this setting, all shares x_i $(1 \leq i \leq n)$ of the secret keys are needed to decrypt any ciphertext, and no individual party is able to decrypt any ciphertext on their own. Notice that the message space $\{0, \cdots, p-1\}$ is quite large compared with the look up table (LUT) range, so we first wrap with coin flipping with the sign kept by choosing a large prime p—which is preferred to be larger than the original message space to en-

sure utility—then distinguish or recover the original message after decryption. During final decryption, everyone checks whether their exponential sum (exponential form of the aggregated weight vector or test result) can be decoded, if the exponential sum of one feature cannot be decoded (overflow the LUT), we can just abandon this feature. No need to restart the protocol because it could pose privacy leakage if one party deliberately adds more noise to overflow the LUT. If this happens, random projection can be used to analyse the impact of the abandoned feature on utility.

5.3 Problem Setup

We focus on training data comprising m training examples stacked as a matrix $\mathbf{X} \in \mathbb{R}_{m \times d}$ on d features and associated labels $\boldsymbol{Y} \in \mathbb{R}_m$. The dataset is partitioned into n disjoint subsets $(\mathbf{X_i}, \boldsymbol{Y_i})$ for $i \in [n]$ representing n parties each curating their own part of the data over a common d-dimensional feature space. The goal of the honest majority of parties is to cooperate to train a joint centroid classifier on this distributed dataset to high utility. In particular we seek no utility loss due to decentralisation. To achieve this goal, each party individually trains local centroid classifiers on their own subset of data. A more accurate global model $f(\mathbf{X}) = \boldsymbol{w}$ can be released by aggregating the parameters of all the local models $f(\mathbf{X_i}) = \boldsymbol{w_i}$. Similarly, the prediction on any test record $\boldsymbol{x}$ can be given by querying the prediction API kept by all the participating parties, which equals the sum of predictions from all the subsets and approximates the result from the combined training set $f(\boldsymbol{x}, \mathbf{X})$.

To achieve strong privacy guarantees, we assume that fewer than $1 - \gamma = 1/3$ of participants are adversarial—the rest are assumed to be honest. A global statistic may be released at a pre-determined level of (ϵ, δ)-differential privacy on the entire dataset; but any other raw or derived local statistic not at this level of DP must be encrypted as required by $(\epsilon, \delta, \gamma)$-distributed differential privacy. *Remark:* [Integrity and availability] Integrity and availability are not our primary goals, however we seek solutions to prevent the most advantageous availability attack whereby one party prevents the others from decoding the response while themself observing the response.

5.4 DPPCC Protocol

In our protocol, we assume that at least a fraction of parties (*e.g.*, 2/3) are honest, while the remaining parties are either honest or malicious. The noise generation task is distributed among the honest parties, who jointly contribute randomness to ensure differential privacy of the global statistic, *i.e.*, aggregated parameters or predictions. Each party may add less noise, and as long as the noise in the global statistic meets the required level, the privacy of the global statistic can be guaranteed. For example, consider the case of sharing weight vectors of individually trained models, *i.e.*, a query function that returns a weight vector $w_i = f(\mathbf{X_i})$ given data $\mathbf{X_i}$. Then the goal is to release a differentially-private global statistic, *i.e.*, noisy sum $\sum_i f(\mathbf{X_i}) + \boldsymbol{r}$, which should be close to the desired global statistic $\sum_i f(\mathbf{X_i})$, and where $\boldsymbol{r}$ is the required noise to guarantee (ϵ, δ)-differential privacy of the global statistic by following Corollary A.2.

However, for our DPPCC protocol, DP alone is not enough to guarantee privacy of local statistics, as only the global statistic is differentially private. Meanwhile, local differential privacy ensures a stronger higher level of privacy at the cost of utility. To preserve privacy and maintain utility, we seek distributed differential privacy leveraging the distributed exponential ElGamal cryptosystem, where the local statistics are encrypted using exponential ElGamal such that each party can only partially decrypt the sum of all the locally released statistics, but cannot access any of them individually.

5.4.1 The Gaussian Mechanism for (ϵ, δ)-DP

We recall the Gaussian mechanism [22] for achieving approximate differential privacy (DDP with a single honest party), with a new analysis leading to privacy guarantees over a wider range of privacy parameters. Namely we remove the constraint $\epsilon < 1$. A detailed explanation and proof is deferred to the supplementary material.

Corollary 5.1. *Adding i.i.d. zero mean and variance σ^2 Gaussian noise to a vector valued statistic on $D \in \mathcal{D}^n$ achieves (ϵ, δ)-DP if $\sigma \geq \frac{\Delta_2 f(\sqrt{2\epsilon + \Phi^{-2}(\delta/2)} - \Phi^{-1}(\delta/2))}{2\epsilon}$, where $\Delta_2 f$ denotes the L_2 sensitivity of the query function f, and Φ is the Gaussian cumulative distribution function (CDF).*

5.4.2 Discrete Gaussian Distribution

As the plaintext in our cryptosystem must be discrete, we require discrete noise. Henceforth, the discrete Gaussian distribution is introduced in Definition A.1.

Definition 5.2 (Discrete Gaussian)**.** *The pmf of the discrete Gaussian is proportional to the pdf of its continuous version. For any $x \in \mathbb{Z}$, it is defined as:*

$$\begin{aligned} P(X = x) \propto f_X(x) &= \mathcal{N}(x; \mu_X, \sigma_X^2) \\ &= \frac{1}{\sqrt{2\pi}\sigma_X} \exp\left(-\frac{(x-\mu_X)^2}{2\sigma_X^2}\right) \end{aligned} \tag{5.1}$$

Our motivation for remaining with the discrete Gaussian distribution is that it is a stable distribution like its continuous version (a sum of discrete Gaussian r.v.'s is still discrete Gaussian), and so will be ideal for realising distributed differential privacy: analysis of overall privacy is made possible by analysing individuals, while also supporting fault tolerance if some individuals are compromised.

Corollary 5.2 (Stability of disc. Gaus.)**.** *The sum of independent discrete Gaussian distributed random variables is discrete Gaussian.*

The corresponding proof for the stability of discrete Gaussian distribution is provided in the supplementary material.

5.4.3 Plaintext Discretisation

Like most cryptographic techniques, our adopted scheme requires plaintext in a integer-valued discrete group. We therefore propose a *Scaling, Rounding, Unscaling* (SRU) Algorithm 5.2 to scale and round the locally computed floating-point statistics.

Algorithm 5.2 Scaling, Rounding, Unscaling (SRU)

Input: scaling factor s**, scalar** x
Scaling (S) is achieved by multiplying by scaling factor s: $x \mapsto sx$.
Rounding (R) converts the scaled real value to the nearest integer for encryption: $sx \mapsto \lfloor sx \rceil$.
Unscaling (U) divides the scaled and rounded scalar by the same scaling factor s: $\lfloor sx \rceil \mapsto \frac{\lfloor sx \rceil}{s}$.

5.4.4 Release Scenarios

Case 1: Sharing Model Parameter

In case 1, each party trains a local model on its own training data, and forms the global model for release by summing over local model parameters. Figure 5.1 depicts the DP-PCC framework for case 1, where each party adds noise $\boldsymbol{r_i}$ to its local model parameters $\boldsymbol{w_i}$ before encryption, then the encrypted noisy model parameters $E(\hat{\boldsymbol{w}}_i)$ are shared among all the parties. The detailed procedure appears in Algorithm 5.3.

Algorithm 5.3 DPPCC protocol for case 1

1: **Training local model**: Each party calculates its weight vector $\boldsymbol{w_i} = f(\mathbf{X_i})$ based on local training data $\mathbf{X_i}$.
2: **Scaling**: Each party multiplies $\boldsymbol{w_i}$ by a specified scaling factor s agreed on by all parties to reduce the effect of rounding on loss, such as $s = 10^3$, $S(\boldsymbol{w_i}) = \boldsymbol{w_i} * s$.
3: **Rounding**: Each party rounds the scaled floating-point $S(\boldsymbol{w_i})$ to integers as $SR(\boldsymbol{w_i}) = R(S(\boldsymbol{w_i}))$.
4: **Adding noise**: Each party adds the scaled discrete Gaussian noise to its scaled and rounded weight vector as $A = SR(\boldsymbol{w_i}) + SR(\boldsymbol{r_i})$.
5: **Encryption**: Each party shares the encrypted $E(A, pk)$ with other parties by encrypting A with the joint public key pk.
6: **Partial decryption**: After each party receives the encrypted noisy weight vectors from other parties, it partially decrypts the sum, then sends the partial decryption back to other parties.
7: **Final decryption and unscaling**: Once each party receives all the partial decryption from all parties, it adds them together as the global weight vector, then divides by s.

For DPPCC in case 1, the released statistic of f is a weight vector, namely, $f(\mathbf{X}) = \boldsymbol{w}$. Assume each item x_{ik} of the whole database $\mathbf{X}$ is bounded by Δ, then the range of each element of the released $\boldsymbol{w}$ is also bounded by Δ, as stated in Theorem 5.1.

Theorem 5.1. *(Query range in case 1). If each item x_{ik} of the whole database $\mathbf{X}$ is bounded by $[-\Delta, \Delta]$, then each item w_k of the non-private query result $f(\mathbf{X}) = \boldsymbol{w}$ is bounded by $[-\Delta, \Delta]$.*

Proof. If each item x_{ik} of the whole database $\mathbf{X}$ is bounded by $[-\Delta, \Delta]$, *i.e.*, $|x_{ik}| \leq \Delta$ for any $i \in \{1, ..., m\}$ and $k \in \{1, ..., d\}$, then the L_1 and L_2 norm of each row vector $\boldsymbol{x_i}$ of the whole database $\mathbf{X}$ are bounded by:

$$\|\boldsymbol{x_i}\|_1 = \sum_{k=1}^{d} |x_{ik}| \leq d\Delta$$
$$\|\boldsymbol{x_i}\|_2 = \sqrt{\sum_{k=1}^{d} x_{ik})^2} \leq \sqrt{d}\Delta$$

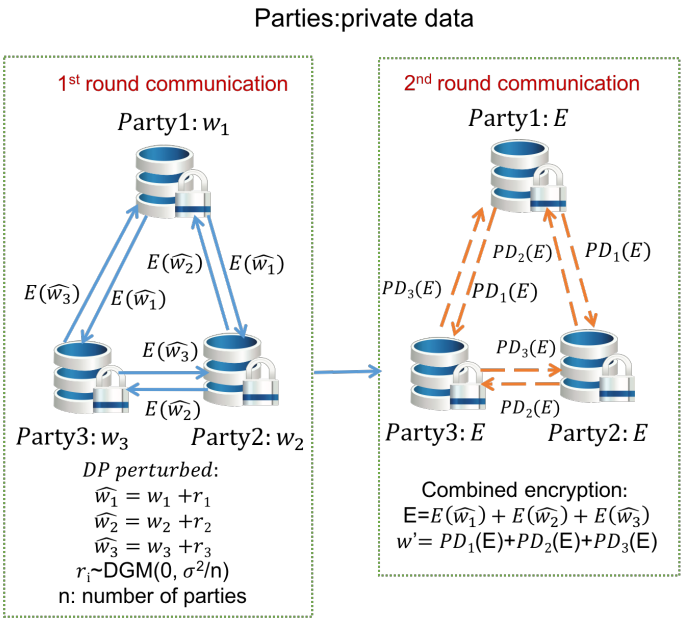


Figure 5.1: DPPCC for case 1.

When $B = 0$, the query result becomes:

$$f(\mathbf{X}) = \frac{1}{2}\left[\frac{(1+B)\sum_{+}\boldsymbol{x_i}}{m_{+1}} - \frac{(1-B)\sum_{-}\boldsymbol{x_i}}{m_{-1}}\right] = \frac{1}{2}\left(\frac{\sum_{+}\boldsymbol{x_i}}{m_{+1}} - \frac{\sum_{-}\boldsymbol{x_i}}{m_{-1}}\right)$$

Hence $w_k = f_k(\mathbf{X}) = \frac{1}{2}\left(\frac{\sum_{+}x_{ik}}{m_{+1}} - \frac{\sum_{-}x_{ik}}{m_{-1}}\right)$, as $|x_{ik}| \leq \Delta$, therefore $|w_k| \leq \frac{1}{2}\left(\left|\frac{\sum_{+}x_{ik}}{m_{+1}}\right| + \left|\frac{\sum_{-}x_{ik}}{m_{-1}}\right|\right) \leq \Delta$. When $B = \frac{m_{+1}-m_{-1}}{m}$, the query result becomes:

$$f(\mathbf{X}) = \frac{1}{2}\left[\frac{(1+B)\sum_{+}\boldsymbol{x_i}}{m_{+1}} - \frac{(1-B)\sum_{-}\boldsymbol{x_i}}{m_{-1}}\right] = \frac{1}{m}\left(\sum_{+}\boldsymbol{x_i} - \sum_{-}\boldsymbol{x_i}\right)$$

Hence $w_k = f_k(\mathbf{X}) = \frac{1}{m}\left(\sum_{+}x_{ik} - \sum_{-}x_{ik}\right)$, as $|x_{ik}| \leq \Delta$, therefore $|w_k| \leq \frac{1}{m}\left(|\sum_{+}x_{ik}| + |\sum_{-}x_{ik}|\right) \leq \Delta$. □

Case 2: Sharing Predictions on Non-Sensitive Test Data

In case 2, an end user sends their test instance to all parties through a secure communication channel. Local models have been trained by each party based on its local training data. Instead of sharing the locally trained weight vectors, local test results are privately

shared among all the parties, who add all the test results together as the global test result, which is sent back to the end user. Figure 5.2 illustrates the DPPCC for case 2, procedures are detailed in Algorithm 5.4.

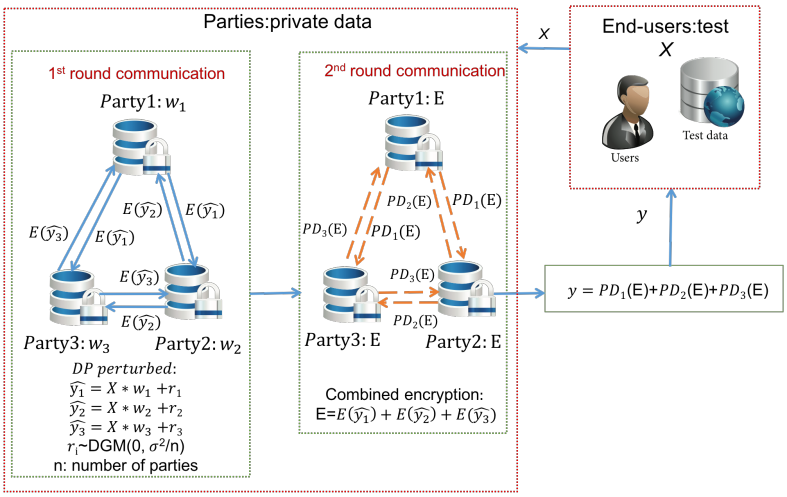


Figure 5.2: DPPCC for case 2.

Algorithm 5.4 DPPCC protocol for case 2

1: **Training local model**: Each party calculates local weight vector $\boldsymbol{w_i} = f(\mathbf{X_i})$ based on local training data $\mathbf{X_i}$.
2: **Sending test data**: For each test record $\boldsymbol{x_i}$ sent by the end user to all parties via secure channels, each party computes local test result y_i.
3: **Scaling**: Multiply y_i by a specified scaling factor s agreed on by all parties to reduce the effect of rounding on loss, such as $s = 10^3$, $S(y_i) = y_i * s$;
4: **Rounding**: Round the scaled floating-point $S(y_i)$ to integer as $SR(y_i) = R(S(y_i))$;
5: **Adding noise to test result**: Each party adds Gaussian noise r_i to its local test result as $A = SR(y_i) + SR(r_i)$.
6: **Encryption**: Each party encrypts noisy test result with the joint public key pk as $E(A, pk)$ and shares it with other parties.
7: **Partial decryption**: After each party sums over the received encrypted noisy test results from all parties, it partially decrypts the sum, and sends the partial decryption back to other parties.
8: **Final decryption and unscaling**: The requested party adds all the received partial decryption together as the global test result, followed by dividing by s, and sending it back to the end user.

Case 3: Sharing Predictions on Sensitive Test Data

Similar to case 2, case 3 also shares the test results, rather than the weight vector. Hence the analysis for sensitivity is similar to case 2. The only difference lies in whether the test

data from end user is sensitive. For highly sensitive data, the end user would prefer to first flip its test data randomly, then distribute its encrypted test data to all parties. The noise addition procedure is the same as in case 2, except that the end user takes part in the encryption and decryption. In particular, the end user generates a key pair, encrypts its flipped and scaled test data with public key, and broadcasts the encrypted test data and public key to all parties. Each party in the decentralized system first multiples the encrypted test data with its local model parameters to get local encrypted prediction, which follows from the MultiByScalar property of any additive homomorphic encryption, for example, the exponential ElGamal. Each party then adds the encrypted local prediction with the scaled discrete Gaussian noise encrypted with the public key of end user, finally shares the encrypted noisy local prediction with other parties. In the second round of communication, each party runs a consensus protocol to ensure everyone gets the same aggregated prediction. After receiving the aggregated prediction from any party, end user decrypts, unscales and flips it back to get the final prediction. DPPCC for case 3 is illustrated in Figure 5.3 and elaborated in Algorithm 5.5.

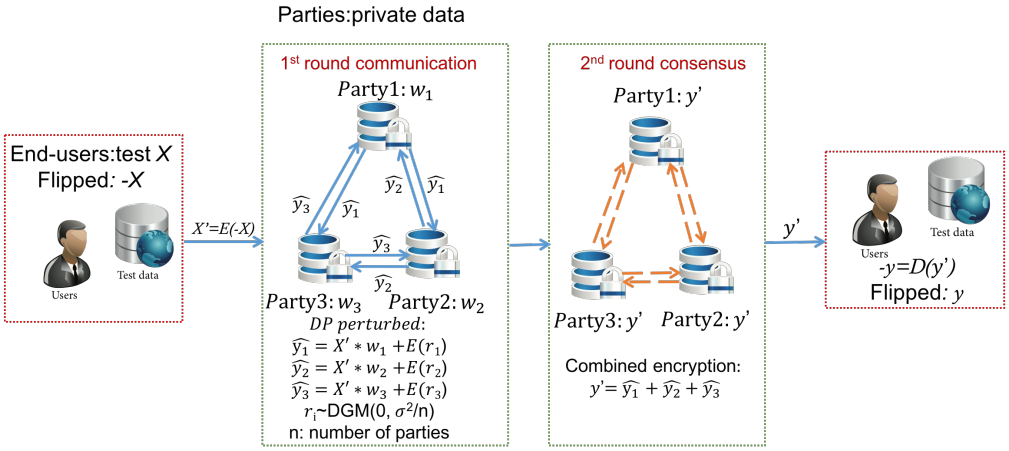


Figure 5.3: DPPCC for case 3.

Theorem 5.2. *(Query range in cases 2 and 3). If each item x_{ik} of the whole database $\mathbf{X}$ is bounded by $[-\Delta, \Delta]$, then the non-private query result $y = f(\boldsymbol{x}, \mathbf{X}) = \boldsymbol{x} * \boldsymbol{w} = \boldsymbol{x} * \sum_{i=1}^{n} \boldsymbol{w_i} = \sum_{i=1}^{n} y_i$ is bounded by $[-d\Delta^2, d\Delta^2]$.*

Proof. If each item of the whole database $\mathbf{X}$ is bounded by $[-\Delta, \Delta]$, then each item of the test record is also bounded by $[-\Delta, \Delta]$, *i.e.*, $|x_k| \leq \Delta$ for any $k \in \{1, ..., d\}$, then the bound

Algorithm 5.5 DPPCC protocol for case 3

1: **Training local model**: Each party trains local weight vector $w_i = f(\mathbf{X_i})$ based on local training data $\mathbf{X_i}$.
2: **End user sends encrypted flipped test record**: End user flips its test record randomly before forwarding it to all parties, then scales and rounds its flipped test record, followed by encrypting with its public key.
3: **Encrypted local prediction**: Each party multiples the encrypted test data with its local model parameters to get encrypted local prediction.
4: **Adding noise**: Each party adds the encrypted local prediction with the discrete Gaussian noise that is scaled and encrypted with the public key of end user.
5: **Aggregation and consensus**: Each party adds all the encrypted noisy local predictions as the aggregated prediction, runs a consensus protocol.
6: **Final decryption**: Once end user receives the aggregated prediction from any party, it decrypts, unscales and flips it back to the final prediction.

for the scalar query result becomes:

$$|y| = |f(\boldsymbol{x}, \mathbf{X})| = |\sum_{k=1}^{d} x_k * w_k| \leq \Delta \sum_{k=1}^{d} |w_k|$$

By Theorem 5.1, $|w_k| \leq \Delta$, therefore, $|y| \leq d\Delta^2$. □

5.5 Theoretical Analysis

5.5.1 Scaling-Rounding-Unscaling (SRU) Loss

Theorem 5.3. *(SRU loss for scalar x). For a specific scaling factor s and any scalar $x \in \mathbb{R}$, the loss of* SRU *algorithm is upper-bounded by $\frac{1}{2s}$.*

Proof. For any scalar $x \in \mathbb{R}$ and scaling factor $s \in \mathbb{R}^+$, the loss can be expressed as: $|x - \frac{\lfloor sx \rceil}{s}| = |\frac{sx - \lfloor sx \rceil}{s}|$. From the definition of rounding to the nearest integer $\lfloor . \rceil$, we know $|sx - \lfloor sx \rceil| \leq \frac{1}{2}$. It then follows $|x - \frac{\lfloor sx \rceil}{s}| \leq \frac{1}{2s}$. □

Since scaling/rounding and unscaling are separate operations that occur at different stages, to be specific, scaling and rounding operations are firstly applied to the original model parameters or test results, while unscaling is applied after the decryption of the noisy sum of the model parameters or test results is derived, hence we explicitly define scaling and rounding operations as *SR* and unscaling operation as *U*. We denote the total loss of *SRU* algorithm as $loss_{SRU}$. The corresponding analysis for the effect of *SRU* on

the global statistic (the sum of the locally released model parameters or test results) is detailed in Lemma 5.1 and Lemma 5.2.

Lemma 5.1. *(SRU loss in case 1). For any weight vector $\boldsymbol{w_i} \in \mathbb{R}^d$ of party i in case 1, the loss of* SRU *algorithm is upper-bounded by $\frac{n\sqrt{d}}{2s}$, where n and d represent the number of parties and the dimension of the released weight vector, s refers to the scaling factor.*

Proof. In case 1, each party firstly applies scaling and rounding to its original weight vector $\boldsymbol{w_i} \in \mathbb{R}^d$, after aggregation and decryption, the sum of all the scaled and rounded noisy weight vectors $SR(\boldsymbol{w_i}) + SR(\boldsymbol{r_i})$ is unscaled by dividing by the scaling factor s. Here, $\boldsymbol{r_i}$ is the discrete Gaussian noise vector added to party i's original weight vector $\boldsymbol{w_i}$, thus $SR(\boldsymbol{w_i})$ needs to add noise $SR(\boldsymbol{r_i}) = \lfloor s\boldsymbol{r_i} \rceil = s\boldsymbol{r_i}$. Therefore,

$$\begin{aligned} loss_{SRU} &= \left\| U\left(\textstyle\sum_{i=1}^{n}(SR(\boldsymbol{w_i}) + SR(\boldsymbol{r_i}))\right) - \textstyle\sum_{i=1}^{n}(\boldsymbol{w_i} + \boldsymbol{r_i}) \right\|_2 \\ &= \left\| \frac{\sum_{i=1}^{n}(\lfloor s\boldsymbol{w_i} \rceil + s\boldsymbol{r_i})}{s} - \textstyle\sum_{i=1}^{n}(\boldsymbol{w_i} + \boldsymbol{r_i}) \right\|_2 \\ &= \left\| \frac{\sum_{i=1}^{n}(\lfloor s\boldsymbol{w_i} \rceil - s\boldsymbol{w_i})}{s} \right\|_2 \end{aligned}$$

From Theorem 5.3, for any $j \in \{1, ..., d\}$, $|sw_{ij} - \lfloor sw_{ij} \rceil| \leq \frac{1}{2}$, it then follows $\left\|s\boldsymbol{w_i} - \lfloor s\boldsymbol{w_i} \rceil\right\|_2 \leq \frac{\sqrt{d}}{2}$, $\left\|\sum_{i=1}^{n}(\lfloor s\boldsymbol{w_i} \rceil - s\boldsymbol{w_i})\right\|_2 \leq \frac{n\sqrt{d}}{2}$. Therefore, $loss_{SRU} \leq \frac{n\sqrt{d}}{2s}$. □

Lemma 5.2. *(SRU loss in case 2). For any test result $y_i \in \mathbb{R}$ of party i in case 2, the loss of* SRU *algorithm is upper-bounded by $\frac{n}{2s}$, where n and s represent the number of parties and scaling factor respectively.*

Proof. In case 2, each party firstly applies scaling and rounding operations to its original test result $y_i \in \mathbb{R}$, after aggregation and decryption, the sum of all the scaled and rounded noisy test results $SR(y_i) + SR(r_i)$ is unscaled by dividing by the scaling factor s. Here, r_i corresponds to the discrete Gaussian noise added to party i's original test result y_i, thus $SR(y_i)$ needs to add noise $SR(r_i) = \lfloor sr_i \rceil = sr_i$. Therefore,

$$
\begin{aligned}
loss_{SRU} &= \left\| U\left(\sum_{i=1}^{n}(SR(y_i)+SR(r_i))\right) - \sum_{i=1}^{n}(y_i+r_i)\right\|_2 \\
&= \left\| \frac{\sum_{i=1}^{n}(\lfloor sy_i \rceil + sr_i)}{s} - \sum_{i=1}^{n}(y_i+r_i)\right\|_2 \\
&= \left\| \frac{\sum_{i=1}^{n}(\lfloor sy_i \rceil - sy_i)}{s}\right\|_2
\end{aligned}
$$

For any y_i, from Theorem 5.3, $|sy_i - \lfloor sy_i \rceil| \leq \frac{1}{2}$, it then follows $\|\sum_{i=1}^{n}(\lfloor sy_i \rceil - sy_i)\|_2 \leq \frac{n}{2}$. Therefore, $loss_{SRU} \leq \frac{n}{2s}$. □

5.5.2 Sensitivity Analysis

Case 1 Sensitivity

For a centroid classifier, the detailed analysis of L_2 sensitivity for case 1 is elaborated in Lemma 5.3 for $B = 0$ and Lemma 5.4 for $B = \frac{m_{+1}-m_{-1}}{m}$, where the total number of records is $m = m_{+1} + m_{-1}$. Changing only *jth* row of $\mathbf{X}$ from $\boldsymbol{x_j}$ to $\boldsymbol{x_j'}$ results in a new database $\mathbf{X}'$ with $f(\mathbf{X}') = \boldsymbol{w}'$, which corresponds to the scenario when exactly one party changes its data by one row. The effect of changing one row on the associated label can be analysed by two cases: change of one row with change to label (without loss of generality, assume the label of the changed row changes from +1 $\rightarrow$ -1); change of row with no change to label (assume the changed row with the original label +1 remains +1).

Lemma 5.3. *(Case 1 sensitivity bound when $B = 0$). Suppose each item x_{ik} of the whole database $\mathbf{X}$ is bounded by Δ, then when $B = 0$, L_{1k} sensitivity of each item of the global weight vector is bounded by $\Delta(\frac{1}{m_{+1}-1} + \frac{1}{m_{-1}+1})$, L_2 sensitivity of the global weight vector is bounded by $\sqrt{d}\Delta(\frac{1}{m_{+1}-1} + \frac{1}{m_{-1}+1})$. The maximum L_{1k} and L_2 sensitivity correspond to $\Delta(1 + \frac{1}{m-1})$ and $\sqrt{d}\Delta(1 + \frac{1}{m-1})$ respectively when m_{+1} takes the boundary points of either 2 or m.*

Proof. When $B = 0$, we consider two cases:

Scenario 1: label is changed

$$
f(\mathbf{X}) = \frac{1}{2}\left(\frac{\sum_{+}\boldsymbol{x_i}}{m_{+1}} - \frac{\sum_{-}\boldsymbol{x_i}}{m_{-1}}\right)
$$

$$f(\mathbf{X}') = \frac{1}{2}\left(\frac{\sum_+ \boldsymbol{x_i} - \boldsymbol{x_j}}{m_{+1}-1} - \frac{\sum_- \boldsymbol{x_i} + \boldsymbol{x_j}'}{m_{-1}+1}\right)$$

$$f(\mathbf{X}) - f(\mathbf{X}') = \frac{1}{2}\left[\left(\frac{\sum_+ \boldsymbol{x_i}}{m_{+1}} - \frac{\sum_- \boldsymbol{x_i}}{m_{-1}}\right) - \left(\frac{\sum_+ \boldsymbol{x_i} - \boldsymbol{x_j}}{m_{+1}-1} - \frac{\sum_- \boldsymbol{x_i} + \boldsymbol{x_j}'}{m_{-1}+1}\right)\right]$$

For simplicity, we split the expression inside the square bracket into two parts: $A_1 = \frac{\sum_+ \boldsymbol{x_i}}{m_{+1}} - \frac{\sum_+ \boldsymbol{x_i} - \boldsymbol{x_j}}{m_{+1}-1}$, $A_2 = \frac{\sum_- \boldsymbol{x_i}}{m_{-1}} - \frac{\sum_- \boldsymbol{x_i} + \boldsymbol{x_j}'}{m_{-1}+1}$

Following the triangle inequality, L_1 and L_2 sensitivity are formulated as:

$$\|f(\mathbf{X}) - f(\mathbf{X}')\|_1 = \frac{1}{2}(\|A_1 - A_2\|_1) \leq \frac{1}{2}(\|A_1\|_1 + \|A_2\|_1)$$

$$\|f(\mathbf{X}) - f(\mathbf{X}')\|_2 = \frac{1}{2}(\|A_1 - A_2\|_2) \leq \frac{1}{2}(\|A_1\|_2 + \|A_2\|_2)$$

Then A_1 and A_2 are analyzed as follows:

$$A_1 = \frac{\sum_+ \boldsymbol{x_i}}{m_{+1}} - \frac{\sum_+ \boldsymbol{x_i} - \boldsymbol{x_j}}{m_{+1}-1} = \frac{-\sum_+ \boldsymbol{x_i} + m_{+1}\boldsymbol{x_j}}{m_{+1}(m_{+1}-1)}$$

$$A_2 = \frac{\sum_- \boldsymbol{x_i}}{m_{-1}} - \frac{\sum_- \boldsymbol{x_i} + \boldsymbol{x_j}'}{m_{-1}+1} = \frac{\sum_- \boldsymbol{x_i} - m_{-1}\boldsymbol{x_j}'}{m_{-1}(m_{-1}+1)}$$

$$\begin{aligned}\|A_{1k}\|_1 = \left\|\frac{-\sum_+ x_{ik} + m_{+1}x_{jk}}{m_{+1}(m_{+1}-1)}\right\|_1 &\leq \frac{\sum_+ \|x_{ik}\|_1 + m_{+1}\|x_{jk}\|_1}{m_{+1}(m_{+1}-1)} \\ \leq \frac{2m_{+1}\Delta}{m_{+1}(m_{+1}-1)} &= \frac{2\Delta}{m_{+1}-1}\end{aligned}$$

$$\begin{aligned}\|A_1\|_2 = \left\|\frac{-\sum_+ \boldsymbol{x_i} + m_{+1}\boldsymbol{x_j}}{m_{+1}(m_{+1}-1)}\right\|_2 &\leq \frac{\sum_+ \|\boldsymbol{x_i}\|_2 + m_{+1}\|\boldsymbol{x_j}\|_2}{m_{+1}(m_{+1}-1)} \\ \leq \frac{2\sqrt{d}m_{+1}\Delta}{m_{+1}(m_{+1}-1)} &= \frac{2\sqrt{d}\Delta}{m_{+1}-1}\end{aligned}$$

$$\|A_{2k}\|_1 = \left\|\frac{1}{m_{-1}(m_{-1}+1)}\left[\sum_- x_{ik} - m_{-1}x'_{jk}\right]\right\|_1$$

$$\leq \frac{1}{m_{-1}(m_{-1}+1)}\left(\left\|\sum_{-} x_{ik}\right\|_1 + m_{-1}\left\|x'_{jk}\right\|_1\right) \leq \frac{2\Delta}{m_{-1}+1}$$

$$\|A_2\|_2 = \left\|\frac{1}{m_{-1}(m_{-1}+1)}\left[\sum_{-}\boldsymbol{x_i} - m_{-1}\boldsymbol{x_j}'\right]\right\|_2$$
$$\leq \frac{1}{m_{-1}(m_{-1}+1)}\left(\left\|\sum_{-}\boldsymbol{x_i}\right\|_2 + m_{-1}\left\|\boldsymbol{x_j}'\right\|_2\right) \leq \frac{2\sqrt{d}\Delta}{m_{-1}+1}$$

Therefore, L_{1k} and L_2 global sensitivity of the query function f follow:

$$\Delta_{1k} f = \|f_k(\mathbf{X}) - f_k(\mathbf{X}')\|_1 \leq \Delta\left(\frac{1}{m_{+1}-1} + \frac{1}{m_{-1}+1}\right)$$

$$\Delta_2 f = \|f(\mathbf{X}) - f(\mathbf{X}')\|_2 \leq \sqrt{d}\Delta\left(\frac{1}{m_{+1}-1} + \frac{1}{m_{-1}+1}\right)$$

Scenario 2: label is not changed

$$f(\mathbf{X}) = \frac{1}{2}\left(\frac{\sum_{+}\boldsymbol{x_i}}{m_{+1}} - \frac{\sum_{-}\boldsymbol{x_i}}{m_{-1}}\right)$$
$$f(\mathbf{X}') = \frac{1}{2}\left(\frac{\sum_{+}\boldsymbol{x_i} - \boldsymbol{x_j} + \boldsymbol{x_j}'}{m_{+1}} - \frac{\sum_{-}\boldsymbol{x_i}}{m_{-1}}\right)$$
$$f(\mathbf{X}) - f(\mathbf{X}') = \frac{1}{2m_{+1}}\left(\boldsymbol{x_j} - \boldsymbol{x_j}'\right)$$
$$\Delta_{1k} f = \|f_k(\mathbf{X}) - f_k(\mathbf{X}')\|_1 \leq \frac{1}{2m_{+1}}\left(\|x_{jk}\|_1 + \|x'_{jk}\|_1\right) \leq \frac{\Delta}{m_{+1}}$$
$$\Delta_2 f = \|f(\mathbf{X}) - f(\mathbf{X}')\|_2 \leq \frac{1}{2m_{+1}}\left(\|\boldsymbol{x_j}\|_2 + \|\boldsymbol{x_j}'\|_2\right) \leq \frac{\sqrt{d}\Delta}{m_{+1}}$$

As $\frac{1}{m_{+1}} < \frac{1}{m_{+1}-1} + \frac{1}{m_{-1}+1}$, we take the bound in the worst case, namely higher bound in scenario 1. To better analyse the sensitivity, we define a specific function as $F = \frac{1}{m_{+1}-1} + \frac{1}{m-m_{+1}+1}$, it can be inferred that the function F of m_{+1} is symmetric with $\frac{m}{2}+1$, and is monotonically increasing when $m_{+1} \geq \frac{m}{2}+1$ and monotonically decreasing when $m_{+1} < \frac{m}{2}+1$, hence it is a convex function with the maximum locating at the boundary points, namely when m_{+1} takes either 2 or m. □ □

Lemma 5.4. *(Case 1 sensitivity bound when $B = \frac{m_{+1}-m_{-1}}{m}$). Suppose each item x_{ik} of the whole database $\mathbf{X}$ is bounded by Δ, then when $B = \frac{m_{+1}-m_{-1}}{m}$, L_{1k} sensitivity of each item of the global*

weight vector is bounded by $\frac{2\Delta}{m}$*, and* L_2 *sensitivity of the global weight vector is bounded by* $\frac{2\sqrt{d}\Delta}{m}$.

Proof. When $B = \frac{m_{+1}-m_{-1}}{m}$, $1+B = \frac{2m_{+1}}{m}$, $1-B = \frac{2m_{-1}}{m}$, similarly, we consider the following two scenarios:

Scenario 1: label is changed

$$B' = \frac{(m_{+1}-1)-(m_{-1}+1)}{m} = \frac{m_{+1}-m_{-1}-2}{m}$$
$$1+B' = \frac{2(m_{+1}-1)}{m}, 1-B' = \frac{2(m_{-1}+1)}{m}$$
$$f(\mathbf{X}) = \frac{1}{2}\left[\frac{(1+B)\sum_{+}\boldsymbol{x_i}}{m_{+1}} - \frac{(1-B)\sum_{-}\boldsymbol{x_i}}{m_{-1}}\right] = \frac{\sum_{+}\boldsymbol{x_i}-\sum_{-}\boldsymbol{x_i}}{m}$$
$$f(\mathbf{X}') = \frac{1}{2}\left[\frac{(1+B')(\sum_{+}\boldsymbol{x_i}-\boldsymbol{x_j})}{m_{+1}-1} - \frac{(1-B')(\sum_{-}\boldsymbol{x_i}+\boldsymbol{x_j}')}{m_{-1}+1}\right] = \frac{1}{m}\left[\left(\sum_{+}\boldsymbol{x_i}-\boldsymbol{x_j}\right)-\left(\sum_{-}\boldsymbol{x_i}+\boldsymbol{x_j}'\right)\right]$$
$$f(\mathbf{X}) - f(\mathbf{X}') = \frac{1}{m}(\boldsymbol{x_j}+\boldsymbol{x_j}')$$
$$\Delta_{1k}f = \|f_k(\mathbf{X}) - f_k(\mathbf{X}')\|_1 \leq \frac{1}{m}\left(\|x_{jk}\|_1 + \|x'_{jk}\|_1\right) \leq \frac{2\Delta}{m}$$
$$\Delta_2 f = \|f(\mathbf{X}) - f(\mathbf{X}')\|_2 \leq \frac{1}{m}\left(\|\boldsymbol{x_j}\|_2 + \|\boldsymbol{x_j}'\|_2\right) \leq \frac{2\sqrt{d}\Delta}{m}$$

Scenario 2: label is not changed

$$B' = B = \frac{m_{+1}-m_{-1}}{m}$$
$$1+B = 1+B' = \frac{2m_{+1}}{m}, 1-B = 1-B' = \frac{2m_{-1}}{m}$$
$$f(\mathbf{X}) = \frac{1}{2}\left[\frac{(1+B)\sum_{+}\boldsymbol{x_i}}{m_{+1}} - \frac{(1-B)\sum_{-}\boldsymbol{x_i}}{m_{-1}}\right] = \frac{\sum_{+}\boldsymbol{x_i}-\sum_{-}\boldsymbol{x_i}}{m}$$
$$f(\mathbf{X}') = \frac{1}{2}\left[\frac{(1+B')(\sum_{+}\boldsymbol{x_i}-\boldsymbol{x_j}+\boldsymbol{x_j}')}{m_{+1}} - \frac{(1-B')\sum_{-}\boldsymbol{x_i}}{m_{-1}}\right] = \frac{1}{m}\left(\sum_{+}\boldsymbol{x_i}-\boldsymbol{x_j}+\boldsymbol{x_j}'-\sum_{-}\boldsymbol{x_i}\right)$$
$$f(\mathbf{X}) - f(\mathbf{X}') = \frac{1}{m}(\boldsymbol{x_j}-\boldsymbol{x_j}')$$
$$\Delta_{1k}f = \|f_k(\mathbf{X}) - f_k(\mathbf{X}')\|_1 \leq \frac{1}{m}\left(\|x_{jk}\|_1 + \|x'_{jk}\|_1\right) \leq \frac{2\Delta}{m}$$
$$\Delta_2 f = \|f(\mathbf{X}) - f(\mathbf{X}')\|_2 \leq \frac{1}{m}\left(\|\boldsymbol{x_j}\|_2 + \|\boldsymbol{x_j}'\|_2\right) \leq \frac{2\sqrt{d}\Delta}{m}.$$

□

Theorem 5.4. *For each local model, if the released local weight vector* $\boldsymbol{w_i} := [w_{i1}, ..., w_{id}]$ *has discrete Gaussian noise added, then the global weight vector* $\boldsymbol{w}$ *that sums over the noisy local weight vectors is* (ϵ, δ)*-differentially private, provided the added discrete Gaussian noise* $\boldsymbol{r_i} :=$

$[r_{i1}, ..., r_{id}]$ follows the randomisation procedure described in Corollary A.2, where $\Delta_2 f$ is the L_2 sensitivity of $f(\mathbf{X}) = \boldsymbol{w}$.

Case 2 and 3 Sensitivity Analysis

Lemma 5.5. *(Case 2 sensitivity bound). Suppose each item x_{ik} of the whole database $\mathbf{X}$ is bounded by Δ, then when B=0, L_2 sensitivity of the global test result is bounded by $d\Delta^2\left(\frac{1}{m_{+1}-1}+\frac{1}{m_{-1}+1}\right)$; when $B=\frac{m_{+1}-m_{-1}}{m}$, L_2 sensitivity of the global test result is bounded by $\frac{2d\Delta^2}{m}$.*

Proof. Cases 2 and 3 aim to release the test result y for any test record $\boldsymbol{x}$ as $f(\boldsymbol{x}, \mathbf{X}) = y$. As test result is a scalar, $\Delta f = \Delta_1 f = \Delta_2 f$. According to Theorem 5.1, both each item w_k of $\boldsymbol{w}$ and each item x_{ik} of the whole database $\mathbf{X}$ are bounded by Δ, therefore, for each test record $\boldsymbol{x}$, and training data $\mathbf{X}$ and $\mathbf{X}'$ that differ in one record, L_1 and L_2 sensitivity of query function f are derived as:

$$\begin{aligned}&\Delta_2 f = \Delta_1 f = \|f(\boldsymbol{x}, \mathbf{X}) - f(\boldsymbol{x}, \mathbf{X}')\|_1 = \|\textstyle\sum_{k=1}^{d} x_k(w_k - w'_k)\|_1\\&\leq \textstyle\sum_{k=1}^{d} \|x_k\|_1 \|w_k - w'_k\|_1 \leq \Delta \sum_{k=1}^{d} \|w_k - w'_k\|_1\end{aligned}$$

When $B = 0$, from Lemma 5.3, $L_{1k} = \|w_k - w'_k\|_1 \leq \Delta\left(\frac{1}{m_{+1}-1}+\frac{1}{m_{-1}+1}\right)$, therefore, it follows:

$$\Delta_2 f = \Delta_1 f \leq d\Delta^2\left(\frac{1}{m_{+1}-1}+\frac{1}{m_{-1}+1}\right)$$

When $B=\frac{m_{+1}-m_{-1}}{m}$, from Lemma 5.4, $L_{1k} = \|w_k - w'_k\|_1 \leq \frac{2\Delta}{m}$, therefore, it follows:

$$\Delta_2 f = \Delta_1 f \leq \frac{2d\Delta^2}{m}.$$ □

Theorem 5.5. *If the released local test result y_i has discrete Gaussian noise added, then the global test result y that sums over all the noisy local test results is (ϵ, δ)-differentially private, provided the added discrete Gaussian noise r_i follows the randomisation procedure described in Corollary A.2, where $\Delta_2 f$ refers to the L_2 sensitivity of $f(\boldsymbol{x}, \mathbf{X}) = y$.*

5.6 Consistency, Fault Tolerance and Availability

5.6.1 Consistency

To prevent dishonest parties sending inconsistent data to honest parties, a consensus protocol that ensures each party is decrypting the same thing is given in Theorem 5.6.

Theorem 5.6. *If honest parties generate noise according to Corollary A.2, then a consensus protocol on what is being decrypted can tolerate k failures, which is less than 1/3 of the total number of parties, and the consensus protocol will terminate in* $k + 1$ *rounds to guarantee consensus.*

5.6.2 Fault Tolerance

The Byzantine faults [122] assume that the participating parties need to agree upon the aggregated model under the constraint that each party may fail: if a single user does not respond in the decryption phase, no decryption can be obtained. The *Byzantine Generals Problem* is solvable only if more than two-thirds of the generals are honest, so a single traitor can confound two honest generals. In particular, with only three generals, no solution exists in the presence of a single traitor. We provide a decentralized interactive solution by removing the trusted server under the assumption that strictly less than one third of the parties are malicious. Specifically, the following solutions can be applied: (i) (t, n)-threshold decryption [123] scheme which requires the cooperation of at least t parties for decryption, if k users fail to send their decryption shares, the (t, n)-threshold property ensures that a decryption can still be computed as long as $k < n - t$. The threshold ElGamal cryptosystem can tolerate the passive corruption of $k < n/3$ parties. (ii) during the noise addition process, each party i perturb $f(\mathbf{x_i})$ by sampling from a Gaussian with mean zero and variance $\frac{3}{2}\sigma/n$, where σ is the lower bound of the noise in Corollary A.2. Afterwards, the perturbed values are shared and summed, yielding $\sum_i f(\mathbf{x_i}) + r$. Since we assume that at least 2/3 of the participants will survive, the total amount of noise would be sufficient, but not excessive.

In the above solution, we are concerned about failures of parties that occur between encryption and final decryption in our protocol. Failures that happen before or after it do not affect the utility of our protocol. Since execution of our protocol takes less than a few

seconds in practice, failures within this small time window are rare and of small size.

5.6.3 Availability

To guarantee the availability of our protocol, no matter when the malicious party drops out, after the first round of communication, all participating parties should check whether their aggregated values are the same. If all the participating parties hold the same value, then the second round of communication will be started; otherwise, rolling back to the first round of communication.

Decoding Availability

As a result of our use of exponential ElGamal to achieve additive homomorphism, it is necessary to construct a decoding table to solve the discrete log of the encoded plaintext. Constructing such a table is only feasible if the size of the table required is small, in relation to the size of the message space. In an honest run of our protocol the size of the table can be predicted and will be within reasonable bounds. However, in a malicious setting there is the potential for an adversary to perform an availability attack by applying a large amount of noise to their value to force the aggregated value outside the range that can be decoded by the constructed table. This results in a successful availability attack, since the honest participants cannot recover the plaintext. Additionally, the attacker would gain a further advantage since they would know the size of the shift and construct their own shifted table to decode the value and remove the additional noise to recover the genuine result. It should be noted that such an attack does not impact on privacy, since the aggregated result is still differentially private.

The solution is to blind the aggregated value prior to decryption. This allows all parties to establish whether the plaintext will be decodable, without revealing the actual plaintext to any adversary. Each party would select at random a blinding factor BF from a suitable range—determined by the size of the constructed table and expected range of aggregate values. BF must be large enough to render a blinded aggregated value useless, but small enough that the sum of the BF from all parties and the aggregate value is within the range that is decodable by the look up table. Each party encrypts their BF as

$E(BF, pk)$ and shares it with the other parties. The sum of the BF's is homomorphically added to the aggregated value and the decryption protocol followed. If the decrypted value is not decodable, *i.e.*, lies outside the lookup table, that repective aggregated value is discarded. If, however, the decrypted value is decodable—by virtue of being in the lookup table—the parties would jointly decrypt the sum of just the BF's, look it up in the lookup table—and subtract the plaintext from the decrypted aggregated value, giving the unblinded aggregated value.

5.7 Performance Evaluation

5.7.1 Baseline Models

Since DPPCC distributes noise among all parties and ensures (ϵ, δ)-differential privacy for the aggregated global statistic, we refer to it here as a global private centroid classifier. We demonstrate the effectiveness of the proposed DPPCC by comparison with the following three baseline models.

Local private centroid classifier excludes cryptosystem as each party adds enough noise to locally released statistic to ensure local differential privacy, so we denote it as local private centroid classifier.

Non-private centroid classifier assumes the same setting as in DPPCC, but is privacy-violating, as it excludes both DP and cryptosystem, the local statistic is directly shared among all parties (maximum utility, minimum privacy).

Standalone centroid classifier excluded cryptosystem as participants train local models on local training data without any collaboration with other parties, then release their private model parameters or test results by adding DP noise (maximum privacy, minimum utility).

5.7.2 Datasets

We investigate the benefits of our DPPCC on three publicly available datasets: Parkinson Speech dataset [124], Breast Cancer Wisconsin dataset [125], and Cardiotocography dataset [126]. The two-class Parkinson Speech dataset consists of 1040 training records,

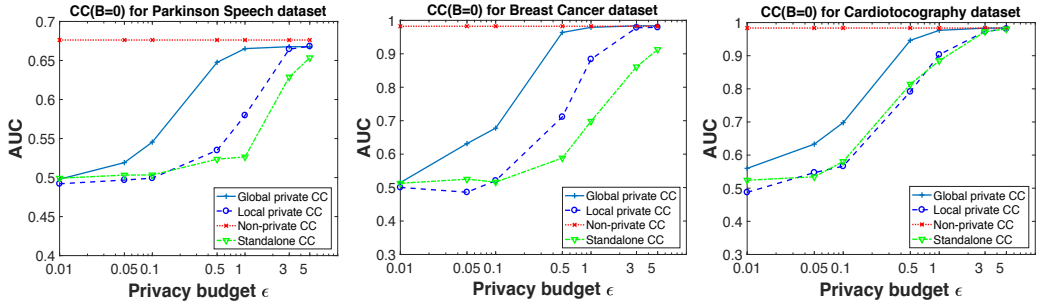
with each record containing 26 features. The Breast Cancer Wisconsin dataset contains two subtypes (malignant and benign), with a total of 569 records, with each record containing 32 features. The 3-class Cardiotocography dataset contains three subtypes, with a total of 2126 records, and each record contains 23 features. Note that these datasets reflect realistic, challenging scenarios for private learning in biomedical domains, in which relatively few records are available. To simulate the situation in which each party constitutes only a subset of the whole database, all the records in the whole database are randomly distributed among different parties such that each party receives nearly the same amount of records. We choose arbitrarily ten parties. All experiments were run based on 50 random splits of data into separate training and test sets, where approximately $2/3$ of the samples from each class were assigned to the training set, while the remaining $1/3$ were assigned to the test set. For classification of the multi-class dataset, we use one-vs-all strategy and average *Area Under the Curve* (AUC) over multiple centroid classifiers.

5.7.3 Experimental Results and Analysis

Four different models are implemented for performance evaluation, namely, global centroid classifier, local private centroid classifier, non-private centroid classifier, and standalone centroid classifier, as stated in Section 5.7.1. We use the metric of *Area Under the Curve* (AUC) for evaluation. The corresponding AUC results for $B = 0$ and unhinged loss in case 1 are presented in Figure 5.4 and Figure 5.5. Similarly, the corresponding AUC results for $B = 0$ and unhinged loss in case 2 are shown in Figure 5.6 and Figure 5.7.

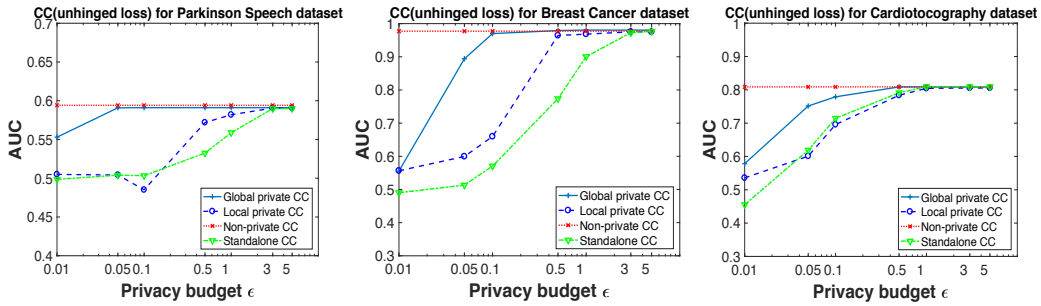
From these figures for both $B = 0$ and unhinged loss in case 1 and case 2 under different privacy budgets ϵ within [0.01,5], it can be observed that: (i) global private centroid classifier (our DPPCC) shows higher AUC than the standalone centroid classifiers, because parties benefit from collaboration in DPPCC; (ii) DPPCC outperforms the local private centroid classifier as less noise is added to locally released statistic, as long as the aggregated noise in the global statistic meets the required level, the privacy of the global statistic can be guaranteed with (ϵ, δ)-DP. In contrast, in local private centroid classifier, each party needs to add the required level of noise (same level as the aggregated noise in DPPCC) to ensure (ϵ, δ)-DP individually, hence much more noise is added to the locally released statistic in local private centroid classifier compared to the noise added to the

locally released statistic in DPPCC; (iii) DPPCC delivers comparable AUC to the non-private centroid classifier in case 1 when $\epsilon \geq 0.5$ for $B = 0$ and $\epsilon \geq 0.1$ for unhinged loss, mainly thanks to collaboration. However, in case 2, the AUC is more inclined to be affected by the added noise than in case 1, as noise is directly added to the test result in case 2, and sensitivity bound is proved to be higher than case 1, as indicated in Lemma 5.3 and Lemma 5.4. Although more noise is added, as expected, case 2 still follows the similar trend as in case 1. In addition, in local private centroid classifier, utility is dependent on the number of parties, because each party adds differentially-private noise to ensure (ϵ, δ)-DP individually, the noise in the aggregation becomes larger with the increasing number of parties, resulting in significant degradation in utility with more parties.



(a) Parkinson Speech. (b) Breast Cancer. (c) Cardiotocography.

Figure 5.4: Centroid classifier(B=0) AUC for all datasets in case 1 ($\delta = 0.05$).

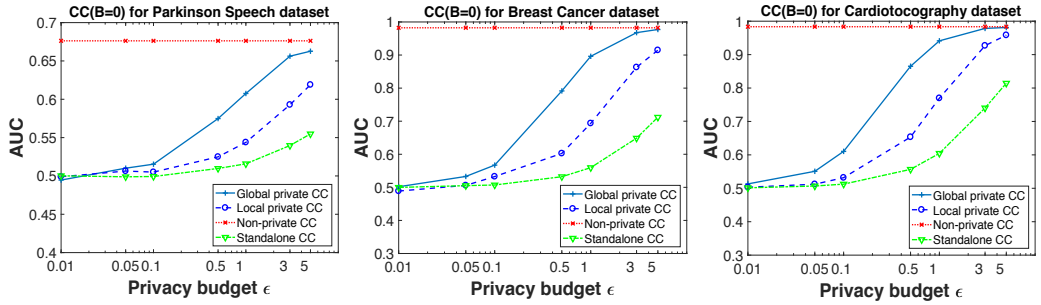


(a) Parkinson Speech. (b) Breast Cancer. (c) Cardiotocography.

Figure 5.5: Centroid classifier(unhinged loss) AUC for all datasets in case 1 ($\delta = 0.05$).

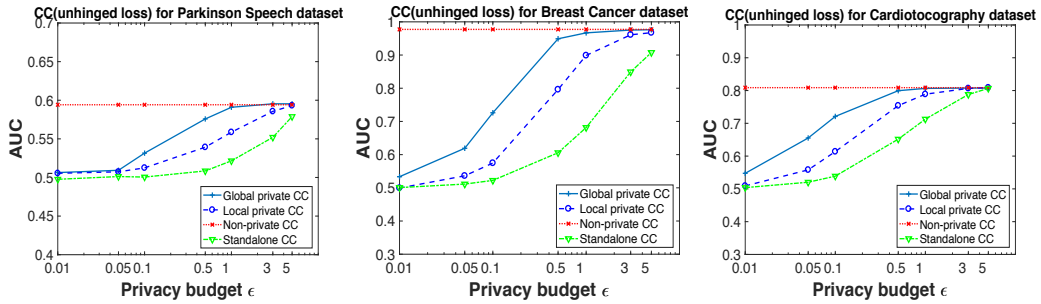
5.7.4 Parameter Selection

Number of parties. The nice theoretical property of the centroid classifier with linear kernel ensures that our scheme's utility is independent of the number of parties.



(a) Parkinson Speech. (b) Breast Cancer. (c) Cardiotocography.

Figure 5.6: Centroid classifier(B=0) AUC for all datasets in case 2 ($\delta = 0.05$).



(a) Parkinson Speech. (b) Breast Cancer. (c) Cardiotocography.

Figure 5.7: Centroid classifier(unhinged loss) AUC for all datasets in case 2 ($\delta = 0.05$).

Scaling factor s. Scaling aims to minimise the effect of rounding on loss. If s is chosen to be 10^N, which means considering N digits to the right of the decimal point; If $s = 1$, then the right of the decimal point is neglected, and the scalar is rounded to the nearest integer; If s is chosen to be less than 1, such as 10^{-N}, which means only N digits to the left of the decimal point is considered. To further explore how to choose s, we take case 1 for analysis, Figure 5.8 shows the corresponding AUC values of DPPCC for $B = 0$ and unhinged loss. It can be figured out that $s = 1$ significantly decreases the AUC compared with $s = 10$, hence s should be chosen to be $s \geq 10$.

5.7.5 Complexity Analysis

Communication cost. Take the Breast Cancer Wisconsin dataset for example, in case 1, instead of sharing the original data matrix m_i*32 (m_i: records of party i, 32: number of features), each party only shares the encrypted noisy weight vector of size 32 with other parties. The number of features can be further reduced without compromising AUC, as

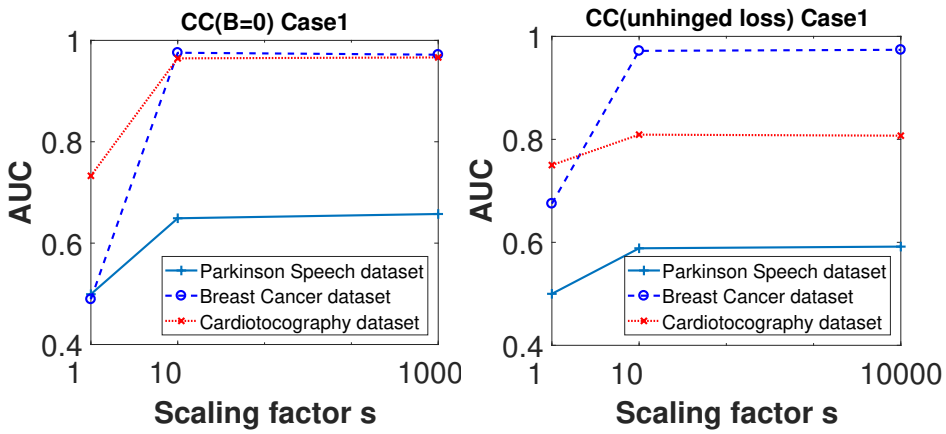


Figure 5.8: DPPCC AUC with varying s in case 1.

manifested in [115]. Similarly, in cases 2 and 3, each party only shares the encrypted noisy test result with other parties. It is apparent that the communication cost of case 2 is lower than that of case 1, as the test result is a scalar.

Computation time. Our distributed ElGamal cryptosystem shown in Section 5.2.3 requires two rounds of communication among all parities. The first round is targeted at forwarding the encryption of each party to all the other parties for partial decryption, and the second round aims to distribute the partial decryption of each party to all the other parties for final decryption. For encryption, running time is dominated by two modular exponentiations. However, these exponentiations are independent of the message, thus can be computed ahead of time. For efficient decryption, we build a look up table (LUT). This only needs to be done once, since the encoding remains stable with successive runs. The limit on the size of the LUT is determined by the available space and time to construct it. Searching is fairly quick because it can be a sorted table. The time spent on different stages for Breast Cancer Wisconsin dataset split among 4 parties with a threshold of 3 is listed in Table 5.1, we set the range of LUT as [-100000, 100000], and average the timings over 50 runs. Timing is proportional to the dimension of the released statistics, but each party can run local training and partial decryption in parallel, hence the number of parties has negligible impact on the timing. Moreover, timings will increase with larger key size, however, it can be implemented in multi-threads. Therefore, our scheme is practical in situations where the plaintext space is small. The encryption mechanism is independent of the classifiers, and it has been verified in [115] that the centroid based approach

is about seven to nine times faster even if the significant time required to search for the optimal C for SVM is neglected.

Table 5.1: Time analysis for Breast Cancer Wisconsin dataset.

Stages	Time (s)
Distributed key generation	1.6
Local model train	1.762
Encryption for one integer	0.055
Decryption for one integer	0.481

Assume we use 2048-bit prime, distributed key generation is run only once, where each party publishes $h_i = g^{x_i}$ of 256 bytes to other parties, which needs around 2 microseconds to be transmitted over a 1 Gbps channel, and costs $256 * (n-1)$ bytes, where n is total number of parties.

For each integer-valued plaintext, each party needs two rounds of communication for encryption and decryption: (i) Each party sends ciphertext (c_1, c_2) of 512 bytes to other parties; and (ii) Each party sends partial decryption of m' of 256 bytes to other parties. Hence, total 768 bytes are required for each integer. It takes around 6.1 microseconds to be transmitted over a 1 Gbps channel. Since parties run distributed key generation, encryption, and decryption in parallel, time complexity is independent on the total number of parties n, in ideal communication channel without delay, for case 1 with d elements, it takes $2 + 6.1 * d$ microseconds. However, as each party needs to transmit message to the remaining $n-1$ parties, the communication complexity is linearly proportional to both the number of elements d and total number of parties n, *i.e.*, the total amount of communication cost is $(256 + 768 * d) * (n-1)$ bytes.

5.8 Summary

In this chapter, we present an efficient decentralized privacy preserving centroid classifier (DPPCC) for three practical cases. It enables parties without significant computational resources to deliver more accurate results from model aggregation or prediction. DPPCC removes the assumption of any third party aggregator—trusted or otherwise—allowing individuals control over their own data. This provides tremendous benefits in highly sensitive application, such as distributed biomedical analysis.

We provide theoretical analysis of the released noisy sum statistic, and introduce stable discrete Gaussian distribution to ensure (ϵ, δ)-differential privacy without any restrictions on ϵ. A distributed exponential ElGamal cryptosystem is developed to mitigate utility loss and guarantee party obliviousness. Preliminary analysis and performance evaluation indicate that DPPCC maintains privacy of each party, while delivers better performance than the local private and standalone centroid classifier, and achieves comparable performance with the non-private centroid classifier.

This page intentionally left blank.

Part II

Privacy-Preserving Data Aggregation

Life is to enjoyed, not endured.

Gordon Hinckley

6

PPFA: Privacy-Preserving Fog-enabled Aggregation in Smart Grid

This chapter focuses on the application of smart metering, where aggregators aim to mine the large amounts of distributed smart meter data that are required for specific purposes, including billing, load modeling/forecasting, power theft detection, pricing services, monitoring of power quality, and load management. A key problem in the aggregation of smart metering is *how to minimize the privacy leakage of the locally released statistics while ensuring high utility*. In privacy-preserving aggregation of smart meter data, we consider honest-but-curious aggregators who aim to make queries on the aggregation of smart meter data of individual users.

6.1 Multi-Level Aggregation

The aggregation of energy consumption data at multiple levels of spatial granularity (neighborhood, subdivision, district, city *etc.*) is essential for monitoring and predicting power consumption, network planning and settlement, allocating and balancing loads and resources, and administering power generation and prices according to demand [9, 10, 18–21]. To derive multi-level privacy-preserving aggregation of smart meter data, we use a more efficient and concentrated Gaussian mechanism to distribute noise generation among parties, thus offering provable differential privacy guarantees of the aggregate statistics on both fog and cloud levels. In addition, to ensure aggregator obliviousness and system robustness, we put forward a two-layer encryption scheme: the first layer applies a one-time pad (OTP) to encrypt individual noisy measurements to achieve aggregator obliviousness, while the second layer uses public-key cryptography for authentication purposes. Our scheme is simple, efficient and practical, it requires only one round of data exchange between an end node (*i.e.*, smart meter), its connected fog node and the cloud if there are no node failures, otherwise, one extra round is needed between an end node, its connected fog node and the trusted third party. In our model, both the fog node and the cloud are assumed to be *honest-but-curious*. They might attempt to discover private information about any user, without deviating from the protocol. The end nodes (*i.e.*, tamper-resistant smart meters) can store keys and perform cryptographic computations but do not trust each other. The noise generation task is distributed amongst end nodes and fog nodes. Based on this assumption, we propose a distributed aggregation model, which is synchronous and non-interactive. Furthermore, our scheme only requires the end nodes to store $O(1)$ values, but does not require any bidirectional communication channels from the aggregator to the end nodes or among end nodes. We consider how an honest-but-curious aggregator can learn desired statistics over multiple parties' data, without compromising any party's privacy.

For efficient multi-level aggregation, one potential approach is to rely on the emerging fog architecture as the basis for in-network aggregation, which corresponds to the sum at the fog nodes or the cloud (e.g. the addition on the plaintext), namely, each fog node computes $f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s) = \sum_{i=1}^{s} \boldsymbol{x}_i$ for s smart meters in its covered area, and the cloud calculates $f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \sum_{i=1}^{n} \boldsymbol{x}_i$ for the total n smart meters. The aggregation

task can be distributed to the intermediate fog nodes, with reasonable extra overhead (e.g. addition and decryption on the ciphertext). The superiority of in-network aggregation in terms of communication and energy consumption are manifested in [18]. In addition, our framework provides different levels of aggregation by leveraging the intermediate fog nodes to forward the fog level aggregation of the downstream end nodes, rather than transmitting numerous sensitive measurements directly to the cloud. Since measurements of end nodes are aggregated at the nearby fog nodes, communication cost is significantly reduced. The energy savings of performing in-network aggregation have been shown to be significant and crucial for energy-constrained sensor networks [18]. In our work, we interchangeably use consumers/users/parties/smart meters/participants/end nodes/sensors to refer to the bottom level smart meters, similarly for aggregator/fog node at the middle level, and cloud/service provider at the top level.

6.2 Aggregation Model in Smart Grid

In general, the models for aggregating statistics from multiple parties fall into the following two categories: *sensor-cloud-via-gateway* and *sensor-fog-cloud*.

- *sensor-cloud-via-gateway*: In a typical IoT network, the cloud collects the encrypted sensor data which are usually forwarded by multi-level gateways. Once the cloud receives the required measurements from all the responding end nodes, it decrypts and derives the desired statistics for further analytics. We refer to this framework as *sensor-cloud-via-gateway*. Obviously, such a framework exhibits several disadvantages. First, in the case of numerous end nodes, as all the measurements are forwarded towards the cloud through multiple gateways, it consumes scarce bandwidth, and unavoidably leads to traffic congestion and heavy loads on the cloud. Hence, it is susceptible to Denial-of-service attack (DoS attack) with incoming traffic flooding, as well as lacking scalability. Furthermore, it incurs high power/energy consumption owing to high volumes of data transmission from end nodes to the remote cloud.

- *sensor-fog-cloud*: The most recent hierarchical in-network data aggregation schemes use less powerful intermediate nodes, *i.e.*, sensor node/gateway to conduct the ex-

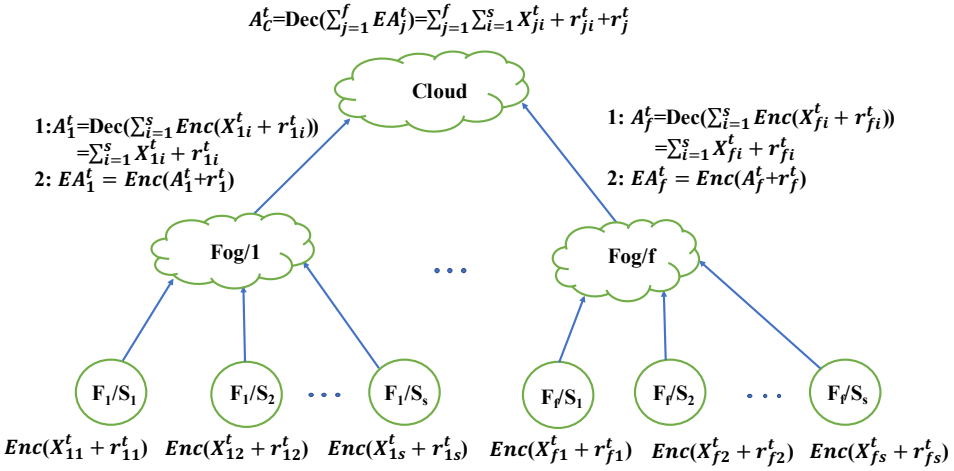


Figure 6.1: *sensor-fog-cloud* aggregation model at time slot t.

pensive encryption and aggregation operations [19]. However, a scalability problem arises as the intermediate nodes cannot support large-scale downstream end nodes, thus limiting its application. To address the resource and scalability issues mentioned above, we instead employ *sensor-fog-cloud* framework, in which fog node takes the role of the intermediate in-network aggregator by connecting with s downstream smart meters. Each sensor meter perturbs its measurement as per sensor level noise, then sends its encrypted noisy measurement to the nearby fog node at each sampling period. Data aggregation and decryption are performed directly on the ciphertext at the nearby fog node, which provides a fine-grained fog level aggregation. Then the decrypted fog level aggregation is further perturbed by the fog level noise and encrypted under the fog node's secret key, then forwarded to the cloud eventually. Cloud level aggregation is derived by aggregating the received fog level aggregation from all the f fog nodes, followed by decryption to obtain the noisy sum of all the measurements of n end nodes.

The *sensor-fog-cloud* aggregation model is illustrated in Figure 6.1. For the readers' convenience, Tables 6.1 contains a list of symbols used in our work.

Table 6.1: Table of symbols.

Symbol	Meaning
Enc	homomorphic encryption function
Dec	homomorphic decryption function
F_j/S_i	sensor i connected to fog node j
Fog/j	fog node j
X_{ji}^t	measurement at F_j/S_i
r_{ji}^t	noise at F_j/S_i
A_j^t	fog level aggregation at Fog/j
EA_j^t	encrypted fog level aggregation at Fog/j
r_j^t	noise at Fog/j
A_C^t	cloud level aggregation at cloud
f	total f fog nodes
s	total s SMs connected to each fog node
n	total $n = s * f$ SMs in the system
t	time slot t
σ_s	noise level required for each sensor
σ_f	noise level required for each fog node

6.3 Privacy-Preserving Fog Aggregation

6.3.1 Privacy Model

We formulate a notion of privacy in smart metering. The traditional differential privacy assumes the presence of a trusted data aggregator who is entitled to see all participants' data in the clear and publish statistics. Our privacy model is stronger in the sense that we do not trust the data aggregator. We ensure that the data aggregator is only able to learn the intended statistics and no additional information. Our scheme protects each individual participant's privacy even when the aggregator has arbitrary auxiliary information (such as public datasets on the web, or through personal knowledge about a specific participant) about a participant's data but has not compromised its secret key, or colludes with a subset of compromised participants, who can arbitrarily reveal their data or noise to the aggregator [10].

Our goal is to guarantee the privacy of each individual's data against an honest-but-curious aggregator, even when the aggregator has arbitrary auxiliary information. At a high level, our formal privacy notion consists of two properties:

- **Aggregator Oblivious**. The aggregator can learn only the noisy sum for each time

slot, and nothing more. Without knowing the aggregator decryption key, one learns nothing about the encrypted data. If the aggregator colludes with a small subset of compromised participants, or if a subset of the encrypted data has been leaked, then the aggregator can inevitably learn the sum of the remaining participants, in this case, the aggregator learns no additional information about the remaining participants' data.

- **Distributed Differential Privacy**. In our model, the participants need not trust the data aggregator or other participants. The required noise in the target statistic is collected from all participants.

Malicious participants that deviate arbitrarily from protocol by modifying, replaying, removing or lying about their values in an attempt to sway the target statistic are out of the scope of this chapter. However, one possible defense is for each participant to use a non-interactive zero-knowledge proof to prove that its encrypted data lies within a valid range, *e.g.*, $\{0, 1, \cdots, \Delta\}$. In this way, each participants's influence is bounded.

6.3.2 Distributed Differential Privacy (DDP)

To achieve (ϵ, δ)-DDP, we employ Dwork and Roth's Gaussian mechanism [22, Theorem A.1] shown in Theorem 2.2. The Gaussian mechanism adds Gaussian noise scaled to $N(0, \sigma^2)$ to each element of the T query responses, where T is the number of time slots. The aggregate function sums over the measurements of a total of n users as a sequence Q of length T, i.e., $f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \sum_{i=1}^{n} \boldsymbol{x}_i$, where $\boldsymbol{x}_i$ represents the measurement of user i in T time slots. At each time slot t, x_i^t $(\forall i \in \{1, \ldots, n\})$ are summed up in a private manner through the DDP scheme, which we denote as $\mathrm{DDP}\left(x_i^t, \frac{\sigma}{\sqrt{n}}\right)$, where σ is the noise required to ensure (ϵ, δ)-differential privacy of the cloud level aggregation in each slot by following Theorem 2.2. These individual noisy values add up to:

$$\hat{x^t} = \sum_{i=1}^{n} \hat{x_i^t} = \sum_{i=1}^{n} (x_i^t + r_i^t) = \sum_{i=1}^{n} x_i^t + r^t. \tag{6.1}$$

For aggregation of time-series data, Shi *et al.*'s scheme [10] provides each participant's differential privacy for every time period, suppose that M^t satisfies ϵ-DP for each $t \in [1, T]$, according to Theorem 2.1, it achieves $T\epsilon$-DP on user-level. Instead, we focus on

providing differential privacy for each participant's daily profile. Because the values of a time-series is a vector, we use L_2 sensitivity of the query function f (sum of all measurements over T time slots) as defined in Definition 2.1. Here, f corresponds to the query function to answer a query sequence with T query responses. Since data can differ in an arbitrary smart meter and the function f corresponds to the sum operation, we choose an empirical value of Δ (domain bound) as the global sensitivity, *i.e.*, worst-case scenario. Then the L_2 sensitivity of f can be expressed as:

$$\Delta_2 f = \sqrt{\sum_{t=1}^{T} \Delta^2}. \tag{6.2}$$

Theorem 6.1. *Given the DDP parameter constraints in Theorem 2.2, and $\Delta_2 f$ in Equation 6.2, the released sum in Equation 6.1 is (ϵ, δ)-differentially private.*

In practice, it seems reasonable that a good estimation for the upper bound is already known. We remark that by using a naturally-weaker form of privacy, *i.e.*, random differential privacy, while replacing worst-case global sensitivity bounds with estimated (actual) sensitivities by assuming data is only bounded with high probability, far superior utility than existing approaches is expected [127, 128].

6.3.3 Encryption Scheme

The DDP procedure described in Section 6.3.2 is not sufficient to guarantee privacy. In order to achieve $O(1)$ error, only the sum (with Gaussian noise $N(0, \sigma^2)$) of individually released statistic is differentially private, but not the individually released statistic (with Gaussian noise $N(0, \frac{\sigma^2}{n})$). The sum $\sum_{i=1}^{n} r_i^t$ meets the requirement for differential privacy, but r_i^t alone does not satisfy differential privacy, thus $x_i^t + r_i^t$ cannot be released directly. To address this problem, we use an additively homomorphic encryption scheme to encrypt the noisy measurement such that the fog node and the cloud can only decrypt the sum of the measurements of its corresponding downstream nodes, but cannot access any of them. Vernam cipher or one-time pad (OTP) has been mathematically proved to be completely secure, it cannot be broken given unbounded ciphertext and time. Therefore, we use OTP for additively homomorphic encryption. OTP uses a keystream of random digits in a similar way to secret sharing. The main idea of forming the ciphertext is to

combine the keystream with the plaintext digits using modular addition one at a time. However, the keystream must be generated completely at random with at least the same length as the plaintext and cannot be used more than once. In addition, the keystream must be kept secure.

In our encryption scheme, a distinct OTP is assumed to be pre-shared between all the participating n sensors, f fog nodes and the cloud through a trusted setup procedure. No further interaction is required except for uploading the encrypted noisy value in each time slot. Each node is configured with a private key and corresponding certificate. The trusted setup can be implemented by a trusted third party or through a standard SMC protocol [10]. For example, each country might have a third party (key management authority) who can generate these certificates and additional "supplier" certificates to supplier companies [129].

The cloud's master key is denoted as k, and the pad pre-shared by each fog node and the cloud is denoted as k_j, such that $\sum_{j=1}^{f} k_j = k$, where $0 < j \leq f$ uniquely identifies a particular fog node. Similarly, the pad pre-shared by each sensor and the corresponding jth fog node is denoted as k_{ji}, such that $\sum_{i=1}^{s} k_{ji} = k_j$, where $0 < i \leq s$ uniquely identifies a particular sensor among a total of s sensors connected with the jth fog node. Therefore, the cloud can only obtain the sum of the fog level aggregation of all the fog nodes, and each fog node can only obtain the sum of the measurements of the connected lower level end nodes. Meanwhile, rather than a XOR operation typically found in stream ciphers, which is insecure under frequency analysis, our encryption scheme uses modular addition (+), which is very well suited for CPU-constrained devices like sensors. In practice, if p = max(x_i), M can be derived as $M = 2^{\lceil log_2(p \times n) \rceil}$.

The OTP scheme only consists of a modular addition operation, thus avoiding the computationally expensive public key cryptosystem. A pseudorandom keystream k can be generated by a secure *pseudo random function* (PRF) by implementing a secure stream cipher, such as Trivium, which is keyed with an end node's secret key k_{ji} and a unique message ID.

Compared with SMC and existing solutions that require several rounds of communication, our protocol merely needs a one-time setup between the cloud, fog nodes and all end nodes. This is beneficial for resource-constrained smart metering system. Moreover,

in our setting all shares k_{ji} ($0 < i \leq s$) of the secret keys are involved to decrypt any ciphertext, thus no SM is able to decrypt any ciphertext alone.

6.3.4 PPFA Procedure

At each time slot t, each fog node computes the sum of the measurements of all the downstream s end nodes (DDP Gaussian noise perturbed input), *i.e.*, $\hat{x_i^t} = x_i^t + \text{DDP}(x_i^t, \sigma_s)$, and the cloud aims to aggregate fog level aggregation of all the fog nodes, following the procedure shown in Algorithm 6.1. As per composition property of DP in Theorem 2.3 and Theorem 2.4, the privacy budget ϵ is allocated to different levels, *i.e.*, ϵ_s (sensor-fog), and ϵ_f (fog-cloud). As transmission from sensor to the fog node is much more sensitive than transmission from fog node to the cloud, hence sensor-fog should be allocated a lower privacy budget. For example, we can add noise as per $\epsilon_s = 0.02$ to the original smart meter data at the bottom level, and noise as per $\epsilon_f = 0.08$ to the middle fog level aggregation.

Algorithm 6.1 PPFA Procedure

Smart meter i: data sanitization and encryption

1: Data sanitization: At each time slot t, SM i that connects to fog node j computes noisy value $x_{ji}^t + \text{DDP}(x_{ji}^t, \sigma_s)$, and rounds it to the nearest integer $\hat{x_{ji}^t} \in [0, M-1]$, where M is a large integer.

2: Encryption: Let k_{ji} be the pad pre-shared by fog node j and SM i, where $k_{ji} \in [0, M-1]$, SM i computes $c_{ji}^t = Enc(\hat{x_{ji}^t}, k_{ji}, M) = \hat{x_{ji}^t} + k_{ji} \pmod M$.

Fog node j: fog level aggregation, decrypt aggregation, aggregation sanitization and encrypt noisy aggregation

1: Fog level aggregation: Derive the encrypted fog level aggregation as $c_j^t = \sum_{i=1}^{s} c_{ji}^t$.

2: Decrypt aggregation: Decrypt the encrypted fog level aggregation as $A_j^t = Dec(c_j^t, k_j, M) = c_j^t - k_j \pmod M$.

3: Aggregation sanitization: Add noise to the decrypted aggregation as $A_j^t + \text{DDP}(A_j^t, \sigma_f)$, and round the noisy value to integer $\hat{A}_j^t \in [0, M-1]$.

4: Encrypt noisy aggregation: Encrypt $\hat{A}_j^t$ as $E\hat{A}_j^t = Enc(\hat{A}_j^t, k_j, M) = \hat{A}_j^t + k_j \pmod M$ and forward it to the cloud.

Cloud: cloud level aggregation, and cloud level decryption

1: Cloud level aggregation: Derive the sum of the encrypted noisy fog level aggregation at the cloud as $c^t = \sum_{j=1}^{f} E\hat{A}_j^t$.

2: Cloud level decryption: For $k = \sum_{j=1}^{f} k_j$, $A_C^t = Dec(c^t, k, M) = c^t - k \pmod M = \sum_{j=1}^{f} \hat{A}_j^t$.

It has been shown that differential privacy is immune to post-processing in [22, Propo-

sition 2.1], therefore, a differentially private query remains differentially private after post-processing [11]. This property allows the fog nodes and the cloud to apply any post-processing function on the output of a differentially private algorithm without diminishing the privacy properties. This property is further exploited in Section 6.5.2 by smoothing the differentially private aggregation of load profiles to improve utility. Theorem 6.2 states that a differentially private aggregation of load profiles remains differentially private after an arbitrary deterministic mapping [11], which is less general compared with an arbitrary randomized mapping in [22, Proposition 2.1]. This theorem ensures that a smoothed differentially private signal with a moving average filter remains differentially private.

Theorem 6.2. *Let $f : \mathbb{R}^n \times \mathbb{R}^T \to \mathbb{R}^T$ be a (ϵ, δ)-differentially private query function with T queries and $g : \mathbb{R}^T \to \mathbb{R}^T$ be an arbitrary deterministic mapping. Then*

$$g \circ f : \mathbb{R}^n \times \mathbb{R}^T \to \mathbb{R}^T$$

is also (ϵ, δ)-differentially private.

6.4 System Robustness

6.4.1 Authenticity

In smart grid, to verify the authenticity of the received message (authentication) and check that the received message is intact (integrity), *i.e.*, ensuing an encrypted message is sent by a legal residential user and has not been altered during the transmission, and preventing unauthorized nodes from injecting fake packets in the networks, we propose a two-layer encryption scheme: the first layer applies OTP to ensure aggregator obliviousness, while the second layer uses public-key cryptography as digital signatures to provide authentication and integrity, which also provides non-repudiation. If the adversary forges or modifies any message, this malicious behaviour should be detected and the message should be abandoned. Therefore, only the correct message can be received by the aggregator for electricity use monitoring. As such, each SM should have a private key and public key pair. Each fog node uses public keys of the connected SMs to authen-

ticate the received messages which are encrypted under the corresponding private keys. Therefore, only the connected fog node who owns the corresponding decryption key can decrypt the received message, and the adversary's malicious behaviours in smart grid communication can be detected.

6.4.2 Node Failure

We have assumed so far that all the s nodes are connected to each fog node, and all the f fog nodes participate in the protocol. However, it might happen that, one or more smart meters fail to participate in a certain time slot for several different reasons (e.g., node or communication failures). This would introduce two effects: first, differential privacy may not be guaranteed any more, since the sum of the noise will not be equivalent to the required level σ. Second, the aggregator will not be able to decrypt the correct aggregate statistics, since $\sum_{i=1}^{s} k_{ji} = k_j$ will not hold any more. Fault tolerance requires the aggregator to be resilient to node failures, *i.e.*, aggregator can still be able to estimate the correct aggregate statistics of the remaining users even when an arbitrary subset of users (unknown in advance) fail. Notice that all the existing fault tolerance schemes introduce either communication/computation/storage complexity or utility loss. Without fault tolerance, Rastogi *et al.* [9] and Shi *et al.* [10] introduce only $O(1)$ error by DDP, while Chan [130] offers fault tolerance at the cost of polylogarithmic error dependent on the number of absent users. In this section, we further extend our scheme to resist node failures by introducing setup extension and sanitization extension.

Setup Extension. One advantage of in-network aggregation is that the failure of any smart meters will only affect the aggregation result of its connected fog node and cloud, without interfering with other SMs and their connected fog nodes. However, the failure can be detected immediately by its connected fog node as its message ID will not be received. Then the failed ID will be reported to the trusted third party, who initializes the trusted setup and issues certificates to all the nodes in the system. To deal with node failure and ensure the correctness of decryption, the trusted third party will update the OTP kept by its connected fog node as the negative sum of the remaining connected nodes, and the pad kept by the cloud to be the negative sum of all the fog nodes. Therefore, the corresponding keys of the non-responding end nodes will not be included during

decryption at the immediate fog node and cloud to ensure the correctness of the fog level and cloud level aggregation. Hence, our solution requires only uni-directional communication, and is much more efficient and simple: it requires setup only once if there are no node failures, otherwise, one extra setup is needed only among the trusted third party, the affected fog node (connected by the failed nodes) and the cloud.

Sanitization Extension. The proposed sanitization extension can resist the failure of up to $n - t$ out of n nodes, where t refers to the number of honest participants. In order to resist the failure of $n - t$ nodes, each node should follow the strategy below to add noise to individual measurements: If at least t honest participants out of a total of n participants participate in the protocol, then we can distribute the noise generation task amongst these t participants. Our construction guarantees that, with high probability, the revealed aggregate statistic meets DP by accumulating noise from the honest participants. Each honest party samples r_i from a Gaussian distribution of standard deviation $\sigma_i = \frac{\sigma}{\sqrt{t}}$ instead of $\frac{\sigma}{\sqrt{n}}$, then the total noise in their sum still satisfies the expected standard deviation σ, *i.e.*, r_i is chosen from a Gaussian distribution that is sufficient to guarantee differential privacy, hence the aggregation remains differentially private. Note that in this case each node may add extra noise to the aggregation in order to ensure differential privacy even if less than $n - t$ nodes fail to send their noise shares to the aggregator. Clearly, this extra noise increases the error if all n nodes operate correctly and add their noise shares faithfully. In the worst case, if every participant believes that the other $n - 1$ participants are disfunctional, each participant would need to add sufficient noise to ensure local differential privacy of its own data, resulting in a large error in the aggregate statistic.

6.4.3 Node Joins and Departures

Our scheme also supports node joins and departures, whenever a participant joins or leaves the system, the trusted key setup phase needs to be restarted among the joining/leaving node, its connected fog node and cloud. To be more specific, when a new node joins, the protocol is restarted by re-issuing the OTP shared between the new node and its connected fog node, and between all the fog nodes and cloud. When a node leaves, it will be excluded and the OTP will be re-issued to its previously connected fog

node and cloud. This scheme is applicable in smart metering scenarios, where participants are relatively static over time.

6.5 Performance Evaluation

In this section, the performance of the proposed PPFA scheme is evaluated in terms of utility analysis, accuracy analysis, communication complexity of *sensor-cloud-via-gateway* and *sensor-fog-cloud* models, and computation complexity of different DP mechanisms.

6.5.1 Utility Analysis

As most cryptosystems work in discrete domain of integers, therefore, we convert floating-point numbers to integers via the *Scaling, Rounding, Unscaling* (SRU) algorithm in Algorithm 5.2. The detailed analysis for the effect of SRU is presented in Lemma 6.1.

Lemma 6.1. *(SRU loss of PPFA). For a specific scaling factor s and perturbed data of party i: $\hat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \boldsymbol{r}_i$, where $\hat{\boldsymbol{x}}_i \in \mathbb{R}^T$. The loss due to the SRU is upper-bounded by $n\sqrt{T}/(2s)$, where n and T refer to the number of smart meters and time slots respectively.*

Proof. Before transmission, each party applies scaling, and rounding on the perturbed data $\hat{\boldsymbol{x}}_i \in \mathbb{R}^T$. After the upper-level fog node/cloud sums over the received data, the aggregate statistics $\sum_i \hat{\boldsymbol{x}}_i$ is unscaled by dividing by the scaling factor s. Hence,

$$\begin{aligned}\text{loss}_{SRU} &= \left\| U\left(\sum_{i=1}^{n} SR(\hat{\boldsymbol{x}}_i)\right) - \sum_{i=1}^{n} \hat{\boldsymbol{x}}_i \right\|_2 \\ &= \left\| \frac{\sum_{i=1}^{n} (\lfloor s\hat{\boldsymbol{x}}_i \rceil)}{s} - \sum_{i=1}^{n} \hat{\boldsymbol{x}}_i \right\|_2 = \left\| \frac{\sum_{i=1}^{n} (\lfloor s\hat{\boldsymbol{x}}_i \rceil - s\hat{\boldsymbol{x}}_i)}{s} \right\|_2 .\end{aligned}$$

Denote $\hat{\boldsymbol{x}}_i := [\hat{x}_{i1}, ..., \hat{x}_{iT}]$, for any $j \in [1, ..., T]$, as $|\lfloor s\hat{x}_{ij} \rceil - s\hat{x}_{ij}| \leq \frac{1}{2}$, hence $\|\lfloor s\hat{\boldsymbol{x}}_i \rceil - s\hat{\boldsymbol{x}}_i\|_2 \leq \frac{\sqrt{T}}{2}$. It then follows $\|\sum_{i=1}^{n} (\lfloor s\hat{\boldsymbol{x}}_i \rceil - s\hat{\boldsymbol{x}}_i)\|_2 \leq \frac{n\sqrt{T}}{2}$. Therefore, $\text{loss}_{SRU} \leq \frac{n\sqrt{T}}{2s}$. □

Theorem 6.3. *For each user, if the released local statistic (smart meter data vector) $\boldsymbol{x_i} \in \mathbb{R}^T$ has DDP Gaussian noise added, then the cloud-level aggregation $\boldsymbol{x}$ that sums over all noisy local statistics is (ϵ, δ)-differentially private, provided that total Gaussian noise $\boldsymbol{r} := (r_1, ..., r_T)$ meets*

variance requirement of Gaussian distribution in Theorem 2.2: $c^2 > 2\ln(1.25/\delta)$, $\sigma \geq c\Delta_2 f/\epsilon$, *where* $0 < \epsilon < 1$, *and* $\Delta_2 f$ *is the* L_2 *sensitivity of* f.

Corollary 6.1. *(PPFA* L_2 *error bound). The error of the cloud-level aggregation of PPFA is higher bounded by* $\sqrt{T}(\frac{n}{2s} + \sigma^2\epsilon/\Delta_2 f - \Delta_2 f/2)$, *where* T *and* n *refer to the number of time slots and number of smart meters,* s *and* $\Delta_2 f$ *denote the scaling factor and* L_2 *sensitivity of* f *respectively.*

Proof. As stated in Lemma 6.1, $loss_{SRU} \leq \frac{n\sqrt{T}}{2s}$, thus the L_2 error bound of the cloud-level aggregation becomes:

$$\begin{aligned} BE &= \left\| U\left(\sum_{i=1}^{n} SR(\hat{\boldsymbol{x}}_i)\right) - \sum_{i=1}^{n} \boldsymbol{x}_i \right\|_2 \\ &= \left\| U\left(\sum_{i=1}^{n} SR(\hat{\boldsymbol{x}}_i)\right) - \sum_{i=1}^{n} (\boldsymbol{x}_i + \boldsymbol{r}_i) + \sum_{i=1}^{n} \boldsymbol{r}_i \right\|_2 \\ &\leq \text{loss}_{SRU} + \left\| \sum_{i=1}^{n} \boldsymbol{r}_i \right\|_2 \\ &\leq \frac{n\sqrt{T}}{2s} + \|\boldsymbol{r}\|_2 \end{aligned}$$

Where $\boldsymbol{r_i}$ is party i's T-dimensional noise vector, and $\boldsymbol{r} = \sum_{i=1}^{n} \boldsymbol{r_i}$ is the total amount of noise required to ensure (ϵ, δ)-DP of the aggregate statistic $\boldsymbol{x} = \sum_{i=1}^{n} \boldsymbol{x}_i$.

As stated in [22, Appendices A], to ensure (ϵ, δ)-DP, it is required that with probability at least $1 - \delta$, each element of $\boldsymbol{r}$ is bounded by $\sigma^2\epsilon/\Delta_2 f - \Delta_2 f/2$. Therefore, with probability at least $1 - \delta$, the following statements hold: $\|\boldsymbol{r}\|_2 \leq \sqrt{T}(\sigma^2\epsilon/\Delta_2 f - \Delta_2 f/2)$, $BE \leq \sqrt{T}(\frac{n}{2s} + \sigma^2\epsilon/\Delta_2 f - \Delta_2 f/2)$. □

6.5.2 Accuracy Analysis

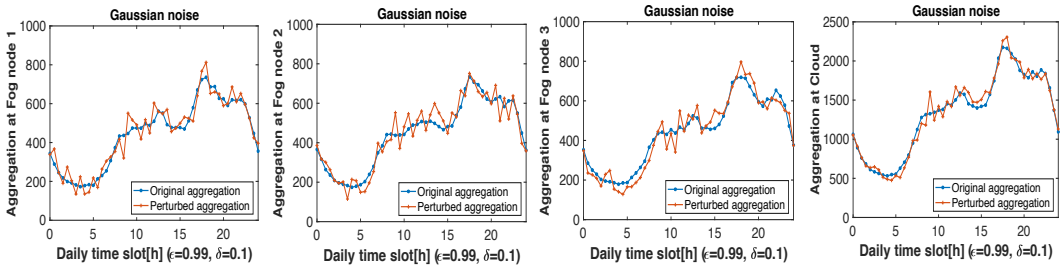


Figure 6.2: Aggregation at different levels for original and (ϵ, δ)-differentially private daily load profiles ($\epsilon = 0.99$, $\delta = 0.1$).

We use the publicly available real-world smart meter dataset from Ireland's Commission for Energy Regulation [131] for evaluation, which consists of 3000 residential homes

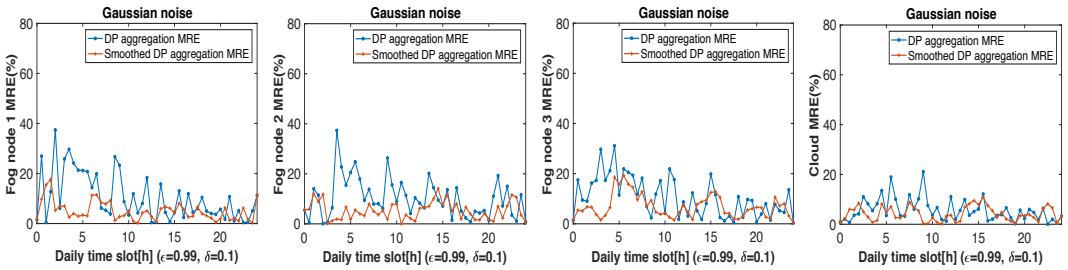


Figure 6.3: MRE at different levels for (ϵ, δ)-DP and smoothed (ϵ, δ)-DP aggregation of daily load profiles ($\epsilon = 0.99$, $\delta = 0.1$).

over 17 months from August 2009 to December 2010. For the experimental implementation, we apply a three-level architecture for evaluation, where the fog level consists of three fog nodes. Each fog node is responsible for the aggregation of 1000 homes at the bottom level, and the cloud aggregates all the fog level aggregations. Electrical load consumption usually exhibits a daily periodicity pattern, and this dataset collects 49 records per day. The average error for each day ($T = 49$ slots) is averaged over both $T = 49$ time slots and 100 runs. The aggregate result is post-processed by smoothing through a moving average filter with a span of 5 to reduce the effect of noise. For sensitivity estimation, we choose Δ as the maximum value of smart meter data each household might spend in each time slot of all daily load profiles. For real-world smart meter daily profiles, the degradation of accuracy is assessed both by visualization and error analysis. The error evaluation for each slot ($T = 1$) and all the T slots are based on the Mean Relative Error (MRE) in percentage defined as below:

$$\text{MRE} = \frac{100}{T}\sum_{t=1}^{T}\left(\frac{\sum_{i=1}^{n}\hat{x}_i^t - \sum_{i=1}^{n}x_i^t}{\sum_{i=1}^{n}x_i^t}\right). \tag{6.3}$$

In Figure 6.2, the dash-blue line shows the original aggregation of daily load profiles at different levels, while the plus-red line corresponds to the aggregation of our proposed PPFA method. It can be observed that PPFA follows consistent trend as the target profile. The corresponding MRE results for DP aggregation and smoothed DP aggregation at different levels over different slots are plotted in Figure 6.3, where after smoothing post-processing, even in the worst case, MRE values over all slots are still lower than 20%.

Figure 6.4 measures MRE over multiple slots under different privacy budget ϵ. The experimental results follow the theoretic analysis, *i.e.*, lower ϵ results in more noise and

higher error, however, after post-processing, even in the worst case in our experimentation ($\epsilon = 0.5$), MRE is still lower than 15%.

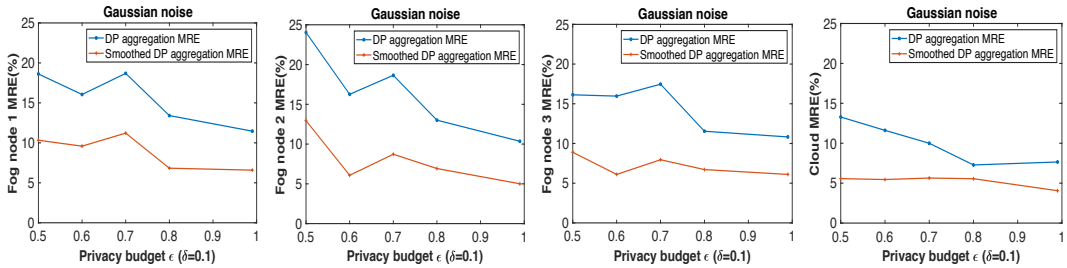


Figure 6.4: MRE at different levels for (ϵ, δ)-DP and smoothed (ϵ, δ)-DP aggregation under different privacy budget ϵ ($\delta = 0.1$).

Furthermore, the effect of the relaxed probability δ is explored in Figure 6.5, which illustrates the MRE over multiple slots under different relaxed probability δ, ranging from [0.001, 0.1]. The experimental results manifest our theoretic analysis, *i.e.*, lower δ tends to be closer to the pure ϵ-DP, leading to more noise and higher error. However, after post-processing, even in the worst case in our experimentation ($\delta = 0.001$), MRE is still lower than 25%.

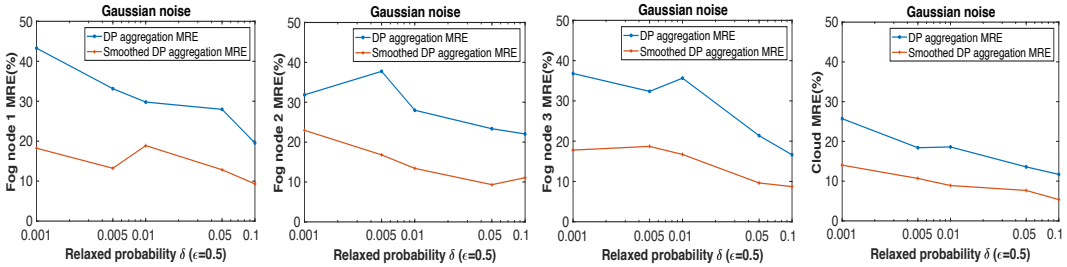


Figure 6.5: MRE at different levels for (ϵ, δ)-DP and smoothed (ϵ, δ)-DP aggregation under different relaxed probability δ ($\epsilon = 0.5$).

To manifest the superiority of DDP, we further study the impact of *Local Differential Privacy* (LDP), where each node adds enough noise to guarantee LDP. Figure 6.6 shows the original aggregation of daily load profiles at different levels, and the aggregation of daily load profiles at different levels that are perturbed by LDP. In comparison with Figure 6.2, it can be clearly observed that LDP significantly degrade the utility as LDP aggregation sums up individual noise values, rendering it largely deviating from the target profile.

To compare with ϵ-DP, we implement the DP scheme used in Ács *et al.* [8], and

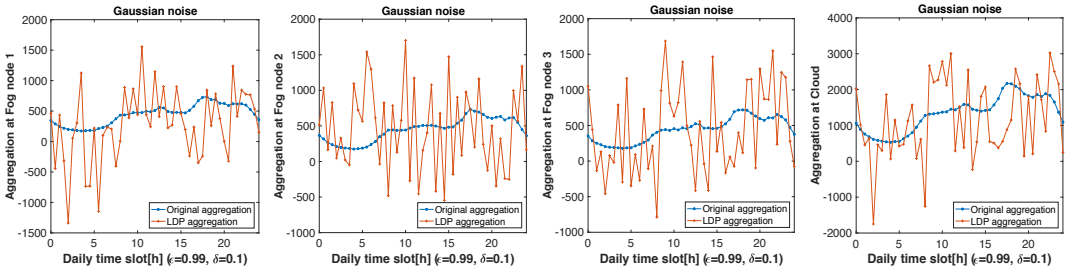


Figure 6.6: Aggregation at different levels for original and (ϵ, δ)-LDP daily load profiles ($\epsilon = 0.99$, $\delta = 0.1$).

Eibl [11], where the divisibility of Laplace distribution is constructed as the sum of i.i.d. gamma distributions. The corresponding MRE results for DP aggregation and smoothed DP aggregation at different levels over all time slots are plotted in Figure 6.7, where MRE is quite high even after smoothing post-processing. Figure 6.8 shows MRE at different levels over all time slots and privacy budget ϵ. After post-processing, even in the best case in our experimentation ($\epsilon = 0.99$), MRE is still higher than 15%. The comparison results demonstrate the superiority of our concentrated Gaussian mechanism, thereby confirming our theoretic analysis.

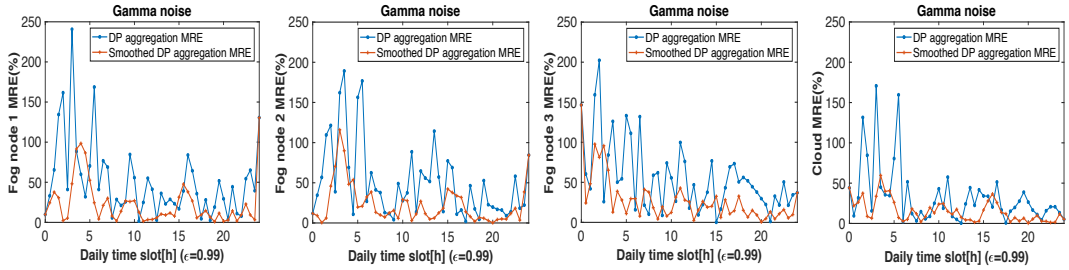


Figure 6.7: MRE at different levels for ϵ-DP and smoothed ϵ-DP aggregation of daily load profiles ($\epsilon = 0.99$).

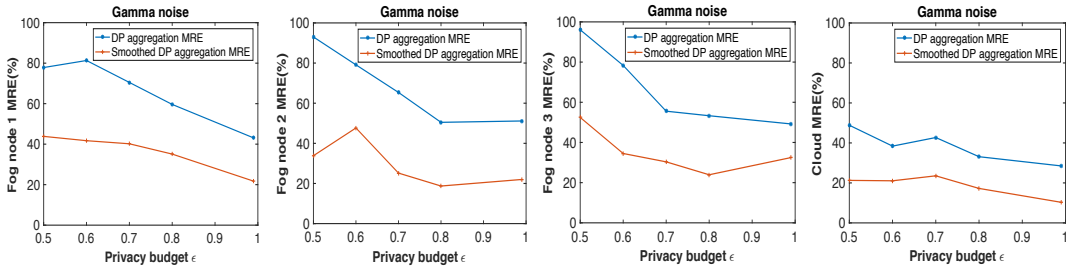


Figure 6.8: MRE at different levels for ϵ-DP and smoothed ϵ-DP aggregation under different privacy budget ϵ.

6.5.3 Complexity Analysis

Communication Complexity. The communication complexity analysis from the perspective of end node level is outlined in Table 6.2. In *sensor-cloud-via-gateway*, end nodes have to transmit large volume of data to the remote cloud via multiple gateways, while in *sensor-fog-cloud*, fog node is one-hop away from end nodes, we denote the transmission distance from end nodes to remote cloud and nearby fog node as d_{sc}, and d_{sf} respectively, where $d_{sf} \ll d_{sc}$. Similarly, as the transmission energy is exponentially proportional to transmission distance, *sensor-cloud-via-gateway* incurs much more energy consumption compared with *sensor-fog-cloud*.

Table 6.2: Communication complexity analysis.

Communication complexity	*sensor-cloud-via-gateway*	*sensor-fog-cloud*
Energy Consumption	$O(e^{d_{sc}})$	$O(e^{d_{sf}})$
Bandwidth Bottleneck	$O(n)$	$O(s)$
Delay	T_{sc}	T_{sf}+T_{fc}

Moreover, in *sensor-cloud-via-gateway*, each end node has to share the bandwidth with all the other $n-1$ nodes, while it reduces to $s-1$ in *sensor-fog-cloud*, where s nodes connect to the same fog node. In terms of delay, the transmission time from end nodes to the cloud and nearby fog node are referred to as T_{sc} and T_{sf} respectively, and T_{fc} is the transmission time from fog node to the cloud, where $T_{sf} \ll T_{sc}$, $T_{fc} \ll T_{sc}$, therefore, $T_{sf} + T_{fc} \ll 2T_{sc}$. It indicates that *sensor-cloud-via-gateway* requires longer time to derive the desired statistics, thus hindering its application in time-critical IoT applications.

Regarding the communication complexity, the results in Table 6.2 demonstrate that *sensor-fog-cloud* performs better, which verifies its benefits in reducing energy cost, extending the battery life, uplifting bandwidth efficiency and achieving timely transmission.

Computation Complexity. To demonstrate the efficiency of our DDP mechanism with Gaussian distribution, we compare with other approaches to DDP, including gamma distribution used in Ács *et al.* [8] and Eibl [11], and geometric distribution used in Shi *et al.* [10]. Table 6.3 shows the time spent on 1000 runs by Matlab R2016a under the parameter settings in Section 6.5.2. It can be seen that Gaussian mechanism offers substantial performance improvement, which is over 450 times faster than both gamma and geomet-

ric distribution. For further detailed comparison of approaches to DDP, the interested reader is referred to the survey paper [132].

Table 6.3: Computation complexity analysis.

DDP mechanism	time (s)
Gaussian	0.5186
Gamma	247.0431
Geometric	268.2539

6.6 Summary

For smart metering, we focus on how to preserve privacy of the load profiles of users. To minimize privacy leakage of the locally released smart meter data and maintain utility with well scaling error, we combine DDP with homomorphic encryption to ensure differential privacy of the aggregate statistic, and aggregator obliviousness, *i.e.*, the honest-but-curious aggregator can periodically collect data and derive aggregate statistics without inferring additional information about any parties.

To facilitate fine-grained aggregation, and consider bandwidth and energy bottlenecks, we explore an efficient and privacy-preserving aggregation model with the aid of fog computing architecture, named PPFA. It enables the intermediate fog nodes to periodically collect data from the connected smart meters and derive fine-grained fog level aggregation for further cloud level aggregation, thus conserving communication energy. To preserve privacy, maintain utility, and ensure system robustness, in addition to using DDP with Gaussian mechanism to guarantee differential privacy of the aggregate statistic, a two-layer encryption scheme is proposed. It combines the efficient stream cipher with public-key cryptography, the only degradation in accuracy is introduced deliberately to satisfy differential privacy. Our privacy model is stronger in the sense that we do not trust the aggregator. We ensure that the aggregator is only able to learn the desired statistics without knowing additional information. Furthermore, our model supports malfunction/non-responding nodes and node joining and leaving. Finally, we give a theoretical analysis for the utility loss, and conduct a comprehensive analysis on the performance of a real-world smart meter dataset, confirming the superiority of our proposed

scheme.

If you wish to succeed, you should use persistence as your good friend, experience as your reference, prudence as your brother and hope as your sentry.

Thomas Edison

7 Conclusion

In this thesis, we first investigate a range of practical privacy-preserving machine learning applications using different frameworks, including collaborative anomaly detection and human activity recognition using centralized framework, and biomedical domain application using decentralized framework. We tackle the security challenges in collaborative anomaly detection with a two-stage scheme called RG+RT to mitigate *maximum a posteriori* (MAP) estimation attacks, and *independent component analysis* (ICA) attack. For privacy-preserving human activity recognition, RG+RP is proposed to resist MAP estimation and ICA attacks, while maintaining model accuracy. These proposed two-stage randomisation schemes are assessed in terms of their recovery resistance to MAP estimation attacks. Preliminary theoretical analysis as well as experimental results on synthetic and real-world datasets demonstrate that both RG+RT and RG+RP exhibit better recovery resistance to MAP estimation attacks than state-of-the-art techniques, meanwhile, high utility is guaranteed. For decentralized privacy-preserving machine learning, we

first examine the practical limitations in the server-based frameworks, then investigate the potential of a decentralized framework, focusing on biomedical domain as motivation. An efficient *Decentralized Privacy-Preserving Centroid Classifier* (DPPCC) is developed for three practical scenarios, where a stable discrete Gaussian mechanism is adapted to realize distributed differential privacy without putting any restriction on the privacy budget ϵ, and a distributed exponential ElGamal cryptosystem is leveraged to maintain utility and ensure party obliviousness. Only the encrypted noisy model parameters or model predictions are shared among all the parties, ensuring that each party learns nothing but the noisy sum of local statistics. The comprehensive experimental study on biomedical datasets demonstrate that our proposed DPPCC outperforms the local private centroid classifier and standalone centroid classifier, and achieves comparable performance to the non-private centroid classifier.

Next, we study how to derive accurate multi-level data aggregation, reduce transmission energy, and preserve user privacy in smart grid application. To this end, we propose a fog-enabled architecture to derive multi-level aggregation of smart meter data. To minimize privacy leakage and mitigate utility, we use a more efficient and concentrated Gaussian mechanism to distribute noise generation among parties, thus offering provable privacy guarantees of the aggregate statistics. In addition, to ensure aggregator obliviousness and system robustness, we put forward a two-layer encryption scheme: the first layer applies a one-time pad (OTP) to encrypt individual noisy measurement in an additively homomorphic manner to achieve aggregator obliviousness, while the second layer uses public-key cryptography to ensure system robustness, including authenticity, node failure, node joins and departures.

We anticipate that our decentralized framework would be beneficial to many open problems at the intersection of machine learning and privacy, including how to facilitate collaboration and ensure privacy and fairness. A number of avenues for further work are attractive. In particular, we would like to further explore the following problems:

1. A major shortcoming of applying differential privacy mechanisms with global sensitivity is that global sensitivity always bounds the worst-case analysis, which would decrease the utility. One alternative is to run privatising mechanisms with lower sensitivity (estimated or actual) by assuming data is only bounded with high

probability, while maintaining (a weaker form of) differential privacy, *i.e.*, random differential privacy [127, 128]. In this way, far superior utility than existing approaches is expected.

2. Robustness against malicious participants in different frameworks is a challenging task, since malicious participants can deviate arbitrarily from protocol by modifying, re-playing, removing or lying about their values. Chapter 5 provides solutions to resist Byzantine faults and availability attack by blinding the aggregated value prior to decryption in the decentralized framework. Other various attacks in other frameworks and corresponding solutions are yet to be explored. For example, federated learning is uniquely vulnerable to attacks that introduce hidden backdoor functionality into the jointly learned global model, *i.e.*, any participant in federated learning can introduce hidden backdoor functionality into the joint global model, *e.g.*, to ensure that an image classifier assigns an attacker-chosen label to images with certain features, or that a word predictor completes certain sentences with an attacker-chosen word [133]. Secure aggregation makes the problem even worse because it prevents the aggregator from auditing the participants' submissions entirely [133]. Another attack in the distributed/federated framework is GAN attack [46], where an adversarial party can intentionally compromise any other party via a parameter server by exploiting the real-time nature of the learning process. During the iterative learning process, the adversarial party can train a GAN to generate prototypical samples of the targeted training data from other parties that was meant to be private. Hence, it's worthwhile to investigate model robustness in the presence of arbitrarily misbehaving participants for future research.

3. The existing frameworks, whether centralized, distributed or decentralized, are all based on the assumption that all the parties contribute equally. In reality, the real-world datasets could be unbalanced and non-independent and identically distributed (non-IID), because some parties may invest a lot of time and effort to harvest high-quality data (for example, in terms of sample quantity, sample distribution over different classes, collecting most informative samples, data cleaning, etc.), while other parties may contribute next to nothing. However, in the end, all the parties can access the same global model or derive similar models no matter how

differently they contribute. This is an essential problem that has been overlooked by far. And this can be an obstacle for the spread of collaborative learning as a new type of powerful learning platform. Building a fair and privacy-preserving learning framework is crucial to make privacy-aware financial or biomedical institutions feel comfortable to join the system and enjoy the benefits brought by data sharing and collaboration. It would be beneficial to build a credibility mechanism to audit participants' contribution without compromising individual privacy and degrading model performance on its main task. Meanwhile, credibility is helpful to detect malicious participants with low credibility, hence isolating these participants. Henceforth, how to design a fair collaborative machine learning system is an important topic for future research.

A

Appendix

A.1 Stability of Discrete Gaussian Distribution

Definition A.1. *(Discrete Gaussian distribution). The probability mass function (pmf) of a discrete Gaussian distribution is proportional to the probability density function (pdf) of its continuous version. The probability mass function for any $X \in \mathbb{Z}$ is therefore defined as:*

$$P(X = x) \propto f_X(x) = \mathcal{N}(x; \mu_X, \sigma_X^2) = \frac{1}{\sqrt{2\pi}\sigma_X} \exp^{-\frac{(x-\mu_X)^2}{2\sigma_X^2}}$$

Corollary A.1. *(Stability of discrete Gaussian distribution). The discrete Gaussian distribution shares the stability property of continuous Gaussians, i.e., the sum of independent discrete Gaussian distributed random variables still follows a discrete Gaussian distribution.*

Proof. If X and Y are independent random variables from two (possibly) different discrete Gaussian distribution, then the distribution of $Z = X + Y$ equals the discrete convolution

of pmfs of X and Y: $P(Z = z) = \sum_{x=-\infty}^{+\infty} P(X = x)P(Y = z - x)$.

Given f_X and f_Y referring to different pdfs for the corresponding continuous Gaussian distributions with

$$P(X = x) \propto f_X(x) = \mathcal{N}(x; \mu_X, \sigma_X^2) = \frac{1}{\sqrt{2\pi}\sigma_X} \exp^{-\frac{(x-\mu_X)^2}{2\sigma_X^2}}, x \in \mathbb{Z}$$

$$P(Y = y) \propto f_Y(y) = \mathcal{N}(x; \mu_Y, \sigma_Y^2) = \frac{1}{\sqrt{2\pi}\sigma_Y} \exp^{-\frac{(y-\mu_Y)^2}{2\sigma_Y^2}}, y \in \mathbb{Z}$$

Substituting into the convolution:

$$P(Z = z) \propto \sum_{x=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_X} \exp\left[-\frac{(x-\mu_X)^2}{2\sigma_X^2}\right] \frac{1}{\sqrt{2\pi}\sigma_Y} \exp\left[-\frac{(z-x-\mu_Y)^2}{2\sigma_Y^2}\right]$$

$$\propto \sum_{x=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sqrt{2\pi}\sigma_X\sigma_Y} \exp\left[-\frac{\sigma_X^2(z-x-\mu_Y)^2 + \sigma_Y^2(x-\mu_X)^2}{2\sigma_X^2\sigma_Y^2}\right]$$

$$\propto \sum_{x=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sqrt{2\pi}\sigma_X\sigma_Y} \exp\left[-\frac{\sigma_X^2(z^2+x^2+\mu_Y^2-2xz-2z\mu_Y+2x\mu_Y) + \sigma_Y^2(x^2+\mu_X^2-2x\mu_X)}{2\sigma_X^2\sigma_Y^2}\right]$$

$$\propto \sum_{x=-\infty}^{\infty} \frac{1}{2\pi\sigma_X\sigma_Y} \exp\left[-\frac{x^2(\sigma_X^2+\sigma_Y^2) - 2x(\sigma_X^2(z-\mu_Y)+\sigma_Y^2\mu_X) + \sigma_X^2(z^2+\mu_Y^2-2z\mu_Y) + \sigma_Y^2\mu_X^2}{2\sigma_X^2\sigma_Y^2}\right]$$

Defining $\sigma_Z = \sqrt{\sigma_X^2 + \sigma_Y^2}$, and completing the square:

$$P(Z = z)$$

$$\propto \sum_{x=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_Z} \frac{1}{\sqrt{2\pi}\frac{\sigma_X\sigma_Y}{\sigma_Z}} \exp\left[-\frac{x^2 - 2x\frac{\sigma_X^2(z-\mu_Y)+\sigma_Y^2\mu_X}{\sigma_Z^2} + \frac{\sigma_X^2(z^2+\mu_Y^2-2z\mu_Y)+\sigma_Y^2\mu_X^2}{\sigma_Z^2}}{2\left(\frac{\sigma_X\sigma_Y}{\sigma_Z}\right)^2}\right]$$

$$\propto \sum_{x=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_Z} \frac{1}{\sqrt{2\pi}\frac{\sigma_X\sigma_Y}{\sigma_Z}} \exp\left[-\frac{\left(x - \frac{\sigma_X^2(z-\mu_Y)+\sigma_Y^2\mu_X}{\sigma_Z^2}\right)^2 - \left(\frac{\sigma_X^2(z-\mu_Y)+\sigma_Y^2\mu_X}{\sigma_Z^2}\right)^2 + \frac{\sigma_X^2(z-\mu_Y)^2+\sigma_Y^2\mu_X^2}{\sigma_Z^2}}{2\left(\frac{\sigma_X\sigma_Y}{\sigma_Z}\right)^2}\right]$$

$$\propto \sum_{x=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_Z} \exp\left[-\frac{\sigma_Z^2\left(\sigma_X^2(z-\mu_Y)^2+\sigma_Y^2\mu_X^2\right)-\left(\sigma_X^2(z-\mu_Y)+\sigma_Y^2\mu_X\right)^2}{2\sigma_Z^2\left(\sigma_X\sigma_Y\right)^2}\right]$$

$$\times \frac{1}{\sqrt{2\pi}\frac{\sigma_X\sigma_Y}{\sigma_Z}} \exp\left[-\frac{\left(x-\frac{\sigma_X^2(z-\mu_Y)+\sigma_Y^2\mu_X}{\sigma_Z^2}\right)^2}{2\left(\frac{\sigma_X\sigma_Y}{\sigma_Z}\right)^2}\right]$$

$$= \frac{1}{\sqrt{2\pi}\sigma_Z} \exp\left[-\frac{(z-(\mu_X+\mu_Y))^2}{2\sigma_Z^2}\right] \sum_{x=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\frac{\sigma_X\sigma_Y}{\sigma_Z}} \exp\left[-\frac{\left(x-\frac{\sigma_X^2(z-\mu_Y)+\sigma_Y^2\mu_X}{\sigma_Z^2}\right)^2}{2\left(\frac{\sigma_X\sigma_Y}{\sigma_Z}\right)^2}\right]$$

The second expression can be written as $f(x) = \sum_{x=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\frac{\sigma_X\sigma_Y}{\sigma_Z}} \exp\left[-\frac{\left(x-\frac{\sigma_X^2(z-\mu_Y)+\sigma_Y^2\mu_X}{\sigma_Z^2}\right)^2}{2\left(\frac{\sigma_X\sigma_Y}{\sigma_Z}\right)^2}\right]$, which can be decomposed into three parts by the mean $\mu = \frac{\sigma_X^2(z-\mu_Y)+\sigma_Y^2\mu_X}{\sigma_Z^2}$: $[-\infty, \mu-1]$, $[\mu]$ and $[\mu+1, \infty]$. As the pdf is monotone on the left and right symmetric part of the mean, the sum of the left and right symmetric parts is approximated by the sum of multiple rectangle areas with interval width 1. According to the monotonic sequence theorem, we can derive that the area under the left $(\sum_{x=-\infty}^{-1} f(x))$ and right $(\sum_{x=1}^{\infty} f(x))$ symmetric part converge to a finite value respectively, *i.e.*, $1/2$, therefore, their sum is upper bounded by the integral of a normal density distribution on continuous real-valued x, which equals to 1. It can be clearly determined from Figure A.1 that each discrete variable x corresponds to a rectangle with the interval width 1, and its pmf value corresponds to the height of the rectangle. Furthermore, as the pdf value of the mean equals to a finite value, the total sum of the three parts converges to a finite value. Therefore, the pmf of z follows:

$$P(Z=z) \propto \frac{1}{\sqrt{2\pi}\sigma_Z} \exp\left[-\frac{(z-(\mu_X+\mu_Y))^2}{2\sigma_Z^2}\right] \qquad \square$$

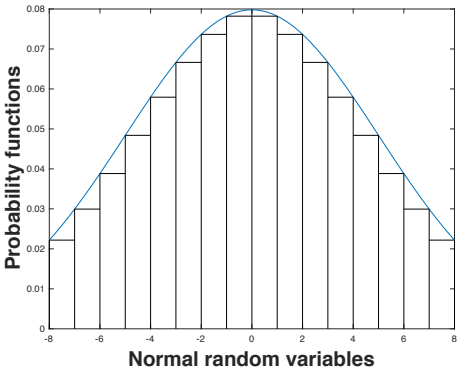


Figure A.1: Blue curve: Probability density function (pdf) of a normal distribution with $\mu = 0, \sigma = 5$; Vertical dark line: Probability mass function (pmf) of a discrete normal distribution with $\mu = 0, \sigma = 5$.

A.2 Bounded Privacy Loss Ensures (ϵ, δ)-Differential Privacy

Let $\mathcal{D}$ be an arbitrary space of record values such as labeled feature vectors, and n be the natural number of records such that $D \in \mathcal{D}^n$ represents a database or dataset. Databases $D, D' \in \mathcal{D}^n$ are said to be neighbouring if they differ on exactly one record. A mechanism $\mathcal{M}$ is a random process indexed by $\mathcal{D}^n$, with random output $\mathcal{M}(D)$ termed a response, governed by probability space[1] $(R, \mathcal{R}, P)$.

In some cases, it is convenient to work with a probability density representing the response distribution of $\mathcal{M}$ (if it exists!). We'll denote by $p_D(x)$ the PDF (formally the Radon-Nikodym derivative) of the response distribution $\mathcal{M}(D)$, indexed again by $D \in \mathcal{D}^n$.

It is common to see researchers define a privacy loss random variable as the absolute log-odds ratio applied to a random response, and define (ϵ, δ)-differential privacy by a high-probability bound on its tail.

Lemma A.1. *Consider mechanism response distribution $\mathcal{M}(D)$ having probability density p_D for all $D \in \mathcal{D}^n$. Define random variable $X = \mathcal{M}(D)$ a response drawn from the mechanism run on D, and a second random variable $\log \frac{p_{D'}(X)}{p_D(X)}$ the privacy loss at X induced by databases*

[1]Formally: R is the range space of $\mathcal{M}$—the space of possible responses; $\mathcal{R}$ is a σ-algebra defining sets of responses with measurable probability; and P a probability measure defined on $\mathcal{R}$, the probability distribution. We'll avoid this level of formality where possible.

$D, D' \in \mathcal{D}^n$. *If for any* $\epsilon > 0$ *and* $0 < \delta < 1$, *any neighbouring* $D, D' \in \mathcal{D}^n$,

$$\Pr\left(\left|\log \frac{p_{D'}(X)}{p_D(X)}\right| > \epsilon\right) \quad \leq \quad \delta \ ,$$

Then $\mathcal{M}$ *preserves* (ϵ, δ)*-differential privacy.*

Proof. Denote by $B_\epsilon = \left\{\left|\log \frac{p_{D'}(X)}{p_D(X)}\right| > \epsilon\right\}$ the event on X that the privacy loss is bounded below by ϵ. Consider any measurable $S \in \mathcal{R}$, then since S is partitioned by $S \cap \overline{B_\epsilon}$ and $S \cap B_\epsilon$,

$$\begin{aligned}
\Pr\left(\mathcal{M}(D) \in S\right) &= \Pr\left(\mathcal{M}(D) \in S \cap \overline{B_\epsilon}\right) + \Pr\left(\mathcal{M}(D) \in S \cap B_\epsilon\right) \\
&= \int_{S \cap \overline{B_\epsilon}} p_D(x)dx + \Pr\left(\mathcal{M}(D) \in S \cap B_\epsilon\right) \\
&\leq \exp(\epsilon) \cdot \int_{S \cap \overline{B_\epsilon}} p_{D'}(x)dx + \Pr\left(\mathcal{M}(D) \in S \cap B_\epsilon\right) \\
&= \exp(\epsilon) \cdot \Pr\left(\mathcal{M}(D') \in S \cap \overline{B_\epsilon}\right) + \Pr\left(\mathcal{M}(D) \in S \cap B_\epsilon\right) \\
&\leq \exp(\epsilon) \cdot \Pr\left(\mathcal{M}(D') \in S\right) + \Pr\left(\mathcal{M}(D) \in B_\epsilon\right) \\
&\leq \exp(\epsilon) \cdot \Pr\left(\mathcal{M}(D') \in S\right) + \delta \ ,
\end{aligned}$$

where the second and third equalities follow from p_D and $p_{D'}$ being densities, the first inequality follows from the privacy loss being bounded above on $S \cap \overline{B_\epsilon}$, the second inequality follows from $S \cap \overline{B_\epsilon} \subseteq S$ and $S \cap B_\epsilon \subseteq B_\epsilon$, and the final inequality follows by assumption that $\Pr(B_\epsilon) \leq \delta$. □

We closely follow the proof of [22, Theorem A.1] to give the lower bound of the standard deviation σ of the Gaussian mechanism that achieves (ϵ, δ)-DP.

Corollary A.2. *If the Gaussian mechanism is run with* $\sigma \geq \frac{(\sqrt{2\epsilon + \left[\Phi^{-1}(\delta/2)\right]^2} - \Phi^{-1}(\delta/2))\Delta_2 f}{2\epsilon}$ *then it achieves* (ϵ, δ)*-DP. Here* $\Delta_2 f$ *denotes the* L_2 *sensitivity of the query function* f, *and* Φ *is the Guassian cumulative distribution function (CDF).*

Proof. Consider running the mechanism $\mathcal{M}$ on a pair of neighbouring databases $D, D' \in \mathcal{D}^n$, such that $\mathcal{M}(D) = f(D) + \boldsymbol{r} = \mathcal{M}(D')$ with $\boldsymbol{r} \sim N(0, \sigma^2 I)$. Let $\boldsymbol{d} = f(D) - f(D')$ be

any vector satisfying $\|\boldsymbol{d}\| \leq \Delta_2 f$, then the privacy loss becomes:

$$\begin{aligned}
&\left|\log \frac{e^{-(1/2\sigma^2)\|\boldsymbol{r}\|^2}}{e^{-(1/2\sigma^2)\|\boldsymbol{r}+\boldsymbol{d}\|^2}}\right| \\
&= \left|\log e^{-(1/2\sigma^2)[\|\boldsymbol{r}\|^2-\|\boldsymbol{r}+\boldsymbol{d}\|^2]}\right| \\
&= \left|\frac{1}{2\sigma^2}(\|\boldsymbol{r}\|^2 - \|\boldsymbol{r}+\boldsymbol{d}\|^2)\right|
\end{aligned}$$

Following [22], for any orthonormal basis $b_1 \dots b_m$ we can write $r_i = \lambda_i b_i$ for $\lambda_i \sim N(0, \sigma^2)$. Without loss of generality, by choosing $b_1 \parallel \boldsymbol{d}$ and by the Pythagorean theorem, [22] obtain the first inequality which is upper bounded by the triangle inequality

$$\begin{aligned}
\left|\frac{1}{2\sigma^2}(\|\boldsymbol{r}\|^2 - \|\boldsymbol{r}+\boldsymbol{d}\|^2)\right| &\leq \left|\frac{1}{2\sigma^2}(-2|\lambda_1|\|\boldsymbol{d}\| - \|\boldsymbol{d}\|^2)\right| \\
&\leq \frac{1}{2\sigma^2}(2|\lambda_1|\Delta_2 f + \Delta_2^2 f).
\end{aligned}$$

To satisfy (ϵ, δ)-DP, we need $\Pr\left(\frac{1}{2\sigma^2}(2|\lambda_1|\Delta_2 f + \Delta_2^2 f\right) \leq \epsilon) > 1 - \delta$. Hence we need,

$$\begin{aligned}
&\Pr\left(|\lambda_1| > \sigma^2\epsilon/\Delta_2 f - \Delta_2 f/2\right) \\
&= 2\Phi\left(\frac{\Delta_2 f/2 - \sigma^2\epsilon/\Delta_2 f}{\sigma}\right) \leq \delta
\end{aligned} \tag{A.1}$$

provided that

$$\sigma^2\epsilon/\Delta_2 f - \Delta_2 f/2 > 0 \Leftrightarrow \sigma^2 > \frac{\Delta_2^2 f}{2\epsilon}. \tag{A.2}$$

Solving (A.1) for σ gives

$$\sigma \geq \frac{\Delta_2 f(\sqrt{2\epsilon + \left[\Phi^{-1}(\delta/2)\right]^2} - \Phi^{-1}(\delta/2))}{2\epsilon} > \frac{\Delta_2 f}{\sqrt{2\epsilon}},$$

where the last inequality follows from $\Phi^{-1}(\frac{\delta}{2}) < 0$ whenever $\delta < 1$. This guarantees condition (A.2).

□

A.3 Weakness of Rastogi *et al.*'s Protocol

We analyse the weakness of Rastogi *et al.*'s protocol based on the following two different attacks:

Attack 1: Collusion attack

Simple Majority Threshold: In the case of a simple majority threshold, the attack requires all dishonest users to collude with a dishonest aggregator. In this scenario, dishonest users and aggregator can run the protocol multiple times, each time with a different honest user. In each run, the partially differentially private value of the honest users will be revealed. For example, we consider the threshold of 2 among three parties A, B, and C: A and B are honest, whilst C is dishonest. The aggregator can collude with C to run the protocol twice, once with A and C and once with B and C. Since C is colluding with the aggregator, he/she can reveal its values and recover the value of the honest party. No party, except C, will run the protocol more than once, so their protocol is not able to detect that something is wrong. Due to the more complex steps, and multiple aggregations, the attack is somewhat all or nothing, in that if a party is to be excluded from a run of the protocol, they need to be excluded throughout, they can't just be excluded during the final aggregation or decryption because some of their r values have been aggregated into early values.

Strictly less than 1/3 dishonest: When a more conventional threshold is selected, the attack becomes harder. It requires an honest party to run the protocol more than once with the same r values. This is a much weaker attack because it is reasonable to assume that users do not run the protocol multiple times.

Attack 2: Sampling attack

In the case of static data, if an untrusted aggregator exits, then sampling attack works on their fault tolerance scheme, which is based on the threshold setting where r_i values are picked from a Laplace distribution or any other decent distribution that dropped off away from zero.

Suppose the same set of honest users can participate over and over again, for example, the aggregator can sample $x_1 + r_1$ arbitrarily many times and get a very good approximation of x_1 (assuming that r_1 is distributed with probability mass function around 0, rather than uniformly from the whole space).

The following procedure illustrates how to get one sample of $x_1 + r_1$:

Step 1: The aggregator implements the first run honestly to learn $\Sigma_u(x_u + r_u)$. It remembers $Enc(x_1 + r_1)$ but cannot decrypt it.

Step 2: The aggregator performs the second run with the same set of parties, who will use the same x_u's but generate new random values, called r'_u. Instead of asking them to decrypt $Enc(\Sigma_u(x_u + r'_u))$ honestly, the aggregator asks instead for a decryption of $Enc(x_1 + r_1 + \Sigma_u(x_u + r'_u))$. In other words, it remembers individual contribution of party 1 from the first run. Decryption now gives the aggregator $x_1 + r_1 + \Sigma_u(x_u + r_u)$, thus $x_1 + r_1$ can be derived by subtracting the output of the second run from the output of the first run.

Now repeat step 1 and step 2 as many times as we like to get lots of samples of $x_1 + r$.

Other Flaws:

1. **Key generation:** Each user generates two keys, a and b by selecting elements at random from the group. However, there is a constraint that sum of all b_i's equals to 0. Then $Enc(a^2)$ is made public as part of "key generation". This part is not explained particularly well, for example, how to generate individual b to conform to a global constraint. In step 4, Algorithm 5.3: Encrypt-Sum-Squared (y_u, r_u) Protocol, the aggregator needs $Enc(a^2)$ to derive c', where $a = \sum_u a_u$, and a_u corresponds to the private key pair $< a_u, b_u >$ of each party, their paper points out $Enc(a^2)$ can be computed and made public in a key generation phase, but not clear who should be responsible for key generation, we assume the key generation in their model is based on a trusted dealer. Moreover, they mentioned $< a_u, b_u >$ can be produced through expensive secret sharing protocols. However, secret sharing is not particularly expensive, verifiable secret sharing is, but they reference Shamir secret sharing, which does not constrain the sum of b_i's to be zero.

2. **Communication complexity:** Another limitation is that their protocol incurs many rounds of communication, such as in the step3, Algorithm 5.4: Encrypt-Noisy-Sum (x_u, r_u), each user communicates with the aggregator for three times, and step4: Encrypt-Sum, each user communicates with the aggregator once, too frequent communications increase the probability of communication failures.

In our model, we make the assumption that honest parties can communicate, and therefore all honest parties will always take part in the first run. It is the existence of

the aggregator that makes these attacks possible in their protocol because each user is unaware of the number of parties it is working with. In our setting, each party is aggregating individually, and therefore they can apply any thresholds themselves. This does incur an increase in communication cost, however, the most likely solution to the attacks in their protocol is to run a consensus protocol to ensure everyone is decrypting the same thing. This is likely to incur additional communication costs.

This page intentionally left blank.

Bibliography

[1] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *ACM SenSys '06 Workshop on World-Sensor-Web: Mobile Device Centric Sensor Networks and Applications*. ACM, 2006.

[2] Deborah Estrin, K Mani Chandy, R Michael Young, Larry Smarr, Andrew Odlyzko, David Clark, Viviane Reding, Toru Ishida, Sharad Sharma, Vinton G Cerf, et al. Participatory sensing: applications and architecture [internet predictions]. *IEEE Internet Computing*, 14(1):12–42, 2010.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. *ACM Sigmod Record*, 29(2), 2000.

[4] Dawud Gordon, Jan-Hendrik Hanne, Martin Berchtold, Ali Asghar Nazari Shirehjini, and Michael Beigl. Towards collaborative group activity recognition using mobile devices. *Mobile Networks and Applications*, 18(3):326–340, 2013.

[5] Michael J McGrath and Cliodhna Ní Scanaill. *Sensor Technologies: Healthcare, Wellness and Environmental Applications*. Apress, 2013.

[6] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genet*, 4(8):e1000167, 2008.

[7] L Sankar, S Raj Rajagopalan, S Mohajer, and HV Poor. Smart meter privacy: A utility-privacy tradeoff framework. *IEEE Trans. Smart Grid*, 3(4):1–10, 2012.

[8] Gergely Ács and Claude Castelluccia. I have a dream!(differentially private smart metering). In *Information hiding*, volume 6958, pages 118–132. Springer, 2011.

[9] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 735–746. ACM, 2010.

[10] Elaine Shi, HTH Chan, Eleanor Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *Annual Network & Distributed System Security Symposium (NDSS)*. Internet Society., 2011.

[11] Günther Eibl and Dominik Engel. Differential privacy for real smart metering data. *Computer Science-Research and Development*, 32(1-2):173–182, 2017.

[12] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[13] Shawn N Murphy, Vivian Gainer, Michael Mendis, Susanne Churchill, and Isaac Kohane. Strategies for maintaining patient privacy in i2b2. *Journal of the American Medical Informatics Association*, 18(Supplement 1):i103–i108, 2011.

[14] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 3–18. IEEE, 2017.

[15] Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *arXiv preprint arXiv:1802.08232*, 2018.

[16] Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A Gunter, and Kai Chen. Understanding membership inferences on well-generalized learning models. *arXiv preprint arXiv:1802.04889*, 2018.

[17] Vehbi C Gungor, Dilan Sahin, Taskin Kocak, Salih Ergut, Concettina Buccella, Carlo Cecati, and Gerhard P Hancke. Smart grid technologies: Communication technologies and standards. *IEEE Transactions on Industrial Informatics*, 7(4):529–539, 2011.

[18] Fengjun Li, Bo Luo, and Peng Liu. Secure information aggregation for smart grids using homomorphic encryption. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 327–332. IEEE, 2010.

[19] Claude Castelluccia, Einar Mykletun, and Gene Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, pages 109–117. IEEE, 2005.

[20] Klaus Kursawe, George Danezis, and Markulf Kohlweiss. Privacy-friendly aggregation for the smart-grid. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 175–191. Springer, 2011.

[21] Zekeriya Erkin and Gene Tsudik. Private computation of spatial and temporal power consumption with smart meters. In *ACNS*, volume 12, pages 561–577. Springer, 2012.

[22] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[23] David Maier. *The theory of relational databases*, volume 11. Computer science press Rockville, 1983.

[24] Olvi L Mangasarian and Edward W Wild. Privacy-preserving classification of horizontally partitioned data via random kernels. In *Proc. International Conference on Data Mining (DMIN)*, volume 2, pages 473–479, 2008.

[25] Murat Kantarcioglu and Chris Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE transactions on knowledge and data engineering*, 16(9):1026–1037, 2004.

[26] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639–644. ACM, 2002.

[27] Jaideep Vaidya and Chris Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215. ACM, 2003.

[28] Jaideep Vaidya and Chris Clifton. Privacy preserving naive bayes classifier for vertically partitioned data. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 522–526. SIAM, 2004.

[29] Geetha Jagannathan and Rebecca N Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 593–599. ACM, 2005.

[30] Jiawei Yuan and Shucheng Yu. Privacy preserving back-propagation neural network learning made practical with cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):212–221, 2014.

[31] Trent McConaghy, Rodolphe Marques, Andreas Müller, Dimitri De Jonghe, Troy McConaghy, Greg McMullen, Ryan Henderson, Sylvain Bellemare, and Alberto Granzotto. Bigchaindb: a scalable blockchain database. *white paper, BigChainDB*, 2016.

[32] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321. ACM, 2015.

[33] Pengtao Xie, Misha Bilenko, Tom Finley, Ran Gilad-Bachrach, Kristin Lauter, and Michael Naehrig. Crypto-nets: Neural networks over encrypted data. *arXiv preprint arXiv:1412.6181*, 2014.

[34] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210, 2016.

[35] Lingjuan Lyu, Yee Wei Law, Sarah M Erfani, Christopher Leckie, and Marimuthu Palaniswami. An improved scheme for privacy-preserving collaborative anomaly

detection. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–6. IEEE, 2016.

[36] Lingjuan Lyu, Xuanli He, Yee Wei Law, and Marimuthu Palaniswami. Privacy-preserving collaborative deep learning with application to human activity recognition. In *Proceedings of the 2017 ACM Conference on Information and Knowledge Management*, pages 1219–1228. ACM, 2017.

[37] Lingjuan Lyu, Yee Wei Law, Jiong Jin, and Marimuthu Palaniswami. Privacy-preserving aggregation of smart metering via transformation and encryption. In *Trustcom/BigDataSE/ICESS, 2017 IEEE*, pages 472–479. IEEE, 2017.

[38] Lingjuan Lyu, James C Bezdek, Yee Wei Law, Xuanli He, and Marimuthu Palaniswami. Privacy-preserving collaborative fuzzy clustering. *Data & Knowledge Engineering*, 116:21–41, 2018.

[39] Martin J Wainwright, Michael I Jordan, and John C Duchi. Privacy aware learning. In *Advances in Neural Information Processing Systems*, pages 1430–1438, 2012.

[40] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.

[41] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Aguera y Arcas. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629*, 2016.

[42] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *ICLR*, 2018.

[43] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 19–38. IEEE, 2017.

[44] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

[45] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2018.

[46] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 603–618. ACM, 2017.

[47] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. ACM, 2017.

[48] Conner Fromknecht, Dragos Velicanu, and Sophia Yakoubov. A decentralized public key infrastructure with identity retention. *IACR Cryptology ePrint Archive*, 2014:803, 2014.

[49] Tsung-Ting Kuo and Lucila Ohno-Machado. Modelchain: Decentralized privacy-preserving healthcare predictive modeling framework on private blockchain networks. *arXiv preprint arXiv:1802.01746*, 2018.

[50] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006.

[51] F Bonomi. Connected vehicles, the internet of things, and fog computing. In *The Eighth ACM International Workshop on Vehicular Inter-Networking (VANET), Las Vegas, USA*, pages 13–15, 2011.

[52] Lingjuan Lyu, Jiong Jin, Sutharshan Rajasegarar, Xuanli He, and Marimuthu Palaniswami. Fog-empowered anomaly detection in iot using hyperellipsoidal clustering. *IEEE Internet of Things Journal*, 4(5):1174–1184, 2017.

[53] Keke Gai, Meikang Qiu, Hui Zhao, Lixin Tao, and Ziliang Zong. Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing. *Journal of Network and Computer Applications*, 59:46–54, 2016.

[54] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[55] Libelium Waspmote Datasheet. 2016. online. `http://www.libelium.com/development/waspmote/documentation/waspmote-datasheet/`. Accessed: 2016-02-14.

[56] Cisco Visual Networking. Cisco global cloud index: Forecast and methodology, 2012-2017. In *CISCO White Paper*, 2013.

[57] Dave Evans. The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011):1–11, 2011.

[58] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security Symposium*, pages 601–618, 2016.

[59] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333. ACM, 2015.

[60] Congzheng Song and Vitaly Shmatikov. The natural auditor: How to tell if someone used your words to train their model. *arXiv preprint arXiv:1811.00513*, 2018.

[61] Ronald Cramer, Ivan Bjerre Damgøard, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015. Cambridge Books Online.

[62] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[63] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[64] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[65] Pascal Paillier et al. Public-key cryptosystems based on composite degree residuosity classes. In *Eurocrypt*, volume 99, pages 223–238. Springer, 1999.

[66] Saghar Estehghari and Yvo Desmedt. Exploiting the client vulnerabilities in internet e-voting systems: Hacking helios 2.0 as an example. *EVT/WOTE*, 10:1–9, 2010.

[67] Yingpeng Sang, Hong Shen, and Hui Tian. Effective reconstruction of data perturbed by random projections. *Computers, IEEE Transactions on*, 61(1):101–117, 2012.

[68] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 99–106. IEEE, 2003.

[69] Zhengli Huang, Wenliang Du, and Biao Chen. Deriving private information from randomized data. In *Proc. 2005 ACM SIGMOD international conference on Management of data*, pages 37–48. ACM, 2005.

[70] Raghu K Ganti, Nam Pham, Yu-En Tsai, and Tarek F Abdelzaher. Poolview: stream privacy for grassroots participatory sensing. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 281–294. ACM, 2008.

[71] Fan Zhang, Li He, Wenbo He, and Xue Liu. Data perturbation with state-dependent noise for participatory sensing. In *INFOCOM, 2012 Proceedings IEEE*, pages 2246–2254. IEEE, 2012.

[72] Keke Chen and Ling Liu. Privacy preserving data classification with rotation perturbation. In *Data Mining, Fifth IEEE International Conference on*, pages 4–pp. IEEE, 2005.

[73] Chris R Giannella, Kun Liu, and Hillol Kargupta. Breaching euclidean distance-preserving data perturbation using few known inputs. *Data & Knowledge Engineering*, 83:93–110, 2013.

[74] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001.

[75] William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[76] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[77] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.

[78] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. A sparse johnson: Lindenstrauss transform. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 341–350. ACM, 2010.

[79] Daniel M Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. *Journal of the ACM (JACM)*, 61(1):4, 2014.

[80] Kun Liu, Hillol Kargupta, and Jessica Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 18(1):92–106, 2006.

[81] Kun Liu, Chris Giannella, and Hillol Kargupta. A survey of attack techniques on privacy-preserving data perturbation methods. In *Privacy-Preserving Data Mining*, pages 359–381. Springer, 2008.

[82] Shibnath Mukherjee, Zhiyuan Chen, and Aryya Gangopadhyay. A privacy-preserving technique for euclidean distance-based mining algorithms using

fourier-related transforms. *The VLDB JournalThe International Journal on Very Large Data Bases*, 15(4):293–315, 2006.

[83] Sarah M Erfani, Yee Wei Law, Shanika Karunasekera, Christopher A Leckie, and Marimuthu Palaniswami. Privacy-preserving collaborative anomaly detection for participatory sensing. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 581–593. Springer, 2014.

[84] Keke Chen, Gordon Sun, and Ling Liu. Towards attack-resilient geometric data perturbation. In *proceedings of the 2007 SIAM international conference on Data mining*, pages 78–89. SIAM, 2007.

[85] Kanishka Bhaduri, Mark D Stefanski, and Ashok N Srivastava. Privacy-preserving outlier detection through random nonlinear data distortion. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 41(1):260–272, 2011.

[86] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[87] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84. ACM, 2007.

[88] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 94–103. IEEE, 2007.

[89] Christos Dimitrakakis, Blaine Nelson, Aikaterini Mitrokotsa, and Benjamin IP Rubinstein. Robust and private bayesian inference. In *International Conference on Algorithmic Learning Theory*, pages 291–305. Springer, 2014.

[90] Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30. ACM, 2009.

[91] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

[92] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 464–473. IEEE, 2014.

[93] Martín Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[94] Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1307–1322. ACM, 2017.

[95] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 51–60. IEEE, 2010.

[96] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. *IEEE Transactions on Information Theory*, 63(6):4037–4049, 2017.

[97] Jack Murtagh and Salil Vadhan. The complexity of computing the optimal composition of differential privacy. In *Theory of Cryptography Conference*, pages 157–175. Springer, 2016.

[98] Cynthia Dwork and Guy N Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.

[99] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[100] Ilya Mironov. Renyi differential privacy. In *Computer Security Foundations Symposium (CSF), 2017 IEEE 30th*, pages 263–275. IEEE, 2017.

[101] Janos Galambos and Italo Simonelli. Products of random variables. *Monographs and Textbooks in Pure and Applied Mathematics*, 268, 2004.

[102] Lingjuan Lyu, Karthik Nandakumar, Benjamin Rubinstein, Jiong Jin, Justin Bedo, and Marimuthu Palaniswami. Ppfa: Privacy preserving fog-enabled aggregation in smart grid. *IEEE Transactions on Industrial Informatics*, 14(8):3733–3744, 2018.

[103] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *ESANN*, 2013.

[104] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408, 2010.

[105] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[106] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[107] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[108] Kun Liu, Chris Giannella, and Hillol Kargupta. An attackers view of distance preserving maps for privacy preserving data mining. In *Knowledge Discovery in Databases: PKDD 2006*, pages 297–308. Springer, 2006.

[109] Kun Liu. *Multiplicative data perturbation for privacy preserving data mining*. PhD thesis, University of Maryland, Baltimore County, 2007.

[110] Nils Y Hammerla, Shane Halloran, and Thomas Ploetz. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880*, 2016.

[111] Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014.

[112] Oresti Banos, Rafael Garcia, Juan A Holgado-Terriza, Miguel Damas, Hector Pomares, Ignacio Rojas, Alejandro Saez, and Claudia Villalonga. Mhealthdroid: a novel framework for agile development of mobile health applications. In *International Workshop on Ambient Assisted Living*, pages 91–98. Springer, 2014.

[113] Lucila Ohno-Machado, Paulo Sérgio Panse Silveira, and Staal Vinterbo. Protecting patient privacy by quantifiable control of disclosures in disseminated databases. *International journal of medical informatics*, 73(7):599–606, 2004.

[114] Deven McGraw. Building public trust in uses of health insurance portability and accountability act de-identified data. *Journal of the American Medical Informatics Association*, 20(1):29–34, 2013.

[115] Justin Bedo, Conrad Sanderson, and Adam Kowalczyk. An efficient alternative to svm based recursive feature elimination with applications in natural language processing and bioinformatics. In *Australasian Joint Conference on Artificial Intelligence*, pages 170–180. Springer, 2006.

[116] Benjamin IP Rubinstein, Peter L Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *arXiv preprint arXiv:0911.5708*, 2009.

[117] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[118] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[119] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[120] Brendan Van Rooyen, Aditya Menon, and Robert C Williamson. Learning with symmetric label noise: The importance of being unhinged. In *Advances in Neural Information Processing Systems*, pages 10–18, 2015.

[121] Yiannis Tsiounis and Moti Yung. On the security of ElGamal based encryption. In *International Workshop on Public Key Cryptography*, pages 117–134. Springer, 1998.

[122] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[123] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 522–526. Springer, 1991.

[124] Betul Erdogdu Sakar, M Erdem Isenkul, C Okan Sakar, Ahmet Sertbas, Fikret Gurgen, Sakir Delil, Hulya Apaydin, and Olcay Kursun. Collection and analysis of a parkinson speech dataset with multiple types of sound recordings. *IEEE Journal of Biomedical and Health Informatics*, 17(4):828–834, 2013.

[125] William H. Wolberg. UCI machine learning repository, 1995.

[126] Diogo Ayres-de Campos, Joao Bernardes, Antonio Garrido, Joaquim Marques-de Sa, and Luis Pereira-Leite. Sisporto 2.0: a program for automated analysis of cardiotocograms. *Journal of Maternal-Fetal Medicine*, 9(5):311–318, 2000.

[127] Rob Hall, Alessandro Rinaldo, and Larry Wasserman. Random differential privacy. *arXiv preprint arXiv:1112.2680*, 2011.

[128] Benjamin IP Rubinstein and Francesco Alda. Pain-free random differential privacy with sensitivity sampling. *arXiv preprint arXiv:1706.02562*, 2017.

[129] Ross Anderson and Shailendra Fuloria. Who controls the off switch? In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 96–101. IEEE, 2010.

[130] T-H Hubert Chan, Elaine Shi, and Dawn Song. Privacy-preserving stream aggregation with fault tolerance. In *International Conference on Financial Cryptography and Data Security*, pages 200–214. Springer, 2012.

[131] Smart metering trial data publication. November 2013. [Online]. `http://www.cer.ie/en/information-centre-reportsandpublications.aspx?article=5dd4bce4-ebd8-475e-b78d-da24e4ff7339`. Accessed: 2017-06-25.

[132] Slawomir Goryczka and Li Xiong. A comprehensive comparison of multiparty secure additions with differential privacy. *IEEE transactions on dependable and secure computing*, 14(5):463–477, 2017.

[133] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.